İSTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

DICTIONARY ENSEMBLE BASED ACTIVE LEARNING FOR MULTIPLE INSTANCE IMAGE CLASSIFICATION

M.Sc. THESIS

Gökhan KOÇYİĞİT

Department of Computer Engineering

Computer Engineering Programme

JUNE 2016

İSTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

DICTIONARY ENSEMBLE BASED ACTIVE LEARNING FOR MULTIPLE INSTANCE IMAGE CLASSIFICATION

M.Sc. THESIS

Gökhan KOÇYİĞİT
504121547

Department of Computer Engineering

Computer Engineering Programme

Thesis Advisor: Asst. Prof. Dr. Yusuf YASLAN

JUNE 2016

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

ÇOKLU ÖRNEKLİ GÖRÜNTÜ SINIFLANDIRMASI İÇİN SÖZLÜK TOPLULUĞU TABANLI AKTİF ÖĞRENME

YÜKSEK LİSANS TEZİ

Gökhan KOÇYİĞİT
504121547

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Tez Danışmanı: Yrd. Doç. Dr. Yusuf YASLAN

HAZİRAN 2016

Gökhan KOÇYIĞIT, a M.Sc. student of İTU Graduate School of Science Engineering and Technology student ID 504121547, successfully defended the thesis/dissertation entitled "DICTIONARY ENSEMBLE BASED ACTIVE LEARNING FOR MULTIPLE INSTANCE IMAGE CLASSIFICATION", which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor : **Asst.Prof.Dr. Yusuf YASLAN** .....
Istanbul Technical University

Jury Members : **Assoc.Prof.Dr. Songül ALBAYRAK** .....
Yıldız Technical University

Assoc.Prof.Dr. Şule Gündüz ÖĞÜDÜCÜ .....
Istanbul Technical University

Date of Submission : 02 May 2016
Date of Defense : 07 June 2016

To my mother, father and my spouse,

FOREWORD

First of all, I would like to thank my supervisor Asst. Prof. Dr. Yusuf YASLAN. Without his guidance and relentless support for my research, I couldn't finish my thesis. He encouraged me and helped me when I encountered difficulties. I also would like to express my gratitude to the Turkish Naval Forces for this opportunity.
Also, I would like to thank my mother and father for supporting me whatever I do. Last, but not least, I thank my spouse for being by my side at all times.

June 2016

Gökhan KOÇYİĞİT
(Lt.J.G.)

TABLE OF CONTENTS

Page

ABBREVIATIONS

MIL	**:** Multiple Instance Learning
MI	**:** Multiple Instance
IS	**:** Instance-Space
BS	**:** Bag Space
ES	**:** Embedded-Space
APR	**:** Axis-Parallel Rectangle
DD	**:** Diverse Density
EM	**:** Expectation-Maximization
EMD	**:** Earth Movers Distance
SVM	**:** Support Vector Machine
DT	**:** Decision Tree
MLP	**:** Multi-Layer Perceptron
NP-Hard	**:** Non-deterministic Polynomial-time Hard
MP	**:** Matching Pursuit
OMP	**:** Orthogonal Matching Pursuit
LARS	**:** Least Angle Regression
MOD	**:** Method of Optimal Directions
SVD	**:** Singular Value Decomposition
SCCE-MIL	**:** Sparse Coding and Classifier Ensemble based Multiple Instance Learning
DEMIAL	**:** Dictionary Ensemble based Multiple Instance Active Learning
KKT	**:** Karush-Kuhn-Tucker conditions
ANN	**:** Artificial Neural Network

SYMBOLS

$\boldsymbol{B_i, B_i^{\{T\}}, B_i^{\{U\}}}$	: The *ith* bag in all dataset, training set and unlabeled set respectively
$\boldsymbol{N, N^{\{T\}}, N^{\{U\}}}$	: Number of examples in all dataset, training set and unlabeled set respectively
$\boldsymbol{K_i}$	: Number of instances in the *ith* bag
$\boldsymbol{C}$	: Number of classes in the dataset
$\boldsymbol{y_i}$	: The *ith* bag label
$\boldsymbol{b_{ij}}$	: The *jth* instance in the *ith* bag
$\boldsymbol{p_{ij}}$	: The probability of the b_{ij} instance in DD algorithm
$\boldsymbol{HD(B_i, B_k)}$	: The minimal Hausdorff distance between two bags
$\boldsymbol{EMD(B_i, B_k)}$	: The Earth Mover Distance between two bags
$\boldsymbol{K_{set}(B_i, B_k)}$	: Set kernel value between two bags
$\boldsymbol{K_{nset}(B_i, B_k)}$	: Normalized set kernel value between two bags
$\boldsymbol{\overline{B}_i}$	: The *ith* bag feature vector
$\boldsymbol{L}$	: Ensemble size
$\boldsymbol{d_{i,j}}$	: Decision of the *ith* classifier of the class *j*
$\boldsymbol{w_i}$	: Weight of the *ith* classifier
$\boldsymbol{\mu_k(x)}$	: Ensemble support value of the class *k*
$\boldsymbol{\widetilde{b}}$	: Instance matrix
$\boldsymbol{M}$	: Number of instances in the instance matrix
$\boldsymbol{D}$	: Dictionary
$\boldsymbol{\alpha_i}$	: Sparse representation of the *ith* instance
$\boldsymbol{J(\alpha)}$	: Regularization function
$\boldsymbol{d_j}$	: The *jth* atom in the dictionary *D*
$\boldsymbol{E_{j_0}}$	: Residual energy of the signal with d_{j_0}
$\boldsymbol{B^*}$	: Queried bag in active learning
$\boldsymbol{P_\theta(y\|B_i)}$	: Posterior probability under the model θ given bag B_i

LIST OF TABLES

Page

LIST OF FIGURES

Page

DICTIONARY ENSEMBLE BASED ACTIVE LEARNING FOR MULTIPLE INSTANCE IMAGE CATEGORIZATION

SUMMARY

In a machine learning problem, each data sample is represented with a feature vector and associated with class information. As an extended version of this approach, in a Multiple Instance Learning (MIL) problem, each data sample consist of a set of feature vectors which may represents the part of the whole entity. Lately, MIL framework draws a lot of attention due to its suitability to the real-world problems e.g. image categorization. In that sense, with the popularity of the internet, it is easy to acquire large collection of data. Using large quantity of data in a MIL setting requires labeling process for each data sample which is a laborious task to accomplish. To overcome this problem, active learning can be used. Active learning is an iterative framework which selects best representative data samples to be labeled by an oracle. In the literature there are several approaches that combine the MIL framework with active learning. Recently sparse coding techniques which approximate a given signal by combining few redundant basis vectors have been applied to MIL framework. In this thesis, we developed an MI active learning method that corporates with sparse coding and classifier ensemble technique called DEMIAL (Dictionary Ensemble based MI Active Learning). The proposed DEMIAL algorithm constructs the classifier ensembles on the sparse feature sets that are obtained from multiple sized dictionaries. The experimental results are obtained on 5 different popular image categorization MIL datasets. Initially we obtained supervised learning results on these datasets with using different base classifiers and ensemble sizes. Then the DEMIAL algorithm is compared with kernel based MI active learning method using two different active learning selection strategies. The effect of the base classifiers and active learning strategies are also considered for DEMIAL algorithm. It is shown that the proposed DEMIAL algorithm performs better than the kernel-based MI active learning method.

ÇOKLU ÖRNEKLİ GÖRÜNTÜ SINIFLANDIRMASI İÇİN SÖZLÜK TOPLULUĞU TABANLI AKTİF ÖĞRENME

ÖZET

Makine öğrenmesi, öğrenilecek verinin geçmiş bilgisini kullanarak modelleyen ve daha sonrasında elde edilecek yeni veriler için karar veren yöntemler bütünüdür. Günümüzde teknolojinin gelişimi ile birlikte çok büyük miktarda veri üretilmekte ve buna bağlı olarak makine öğrenmesi ile bu veriler ve bunlara bağlı problemlerin çözümü için çok çok sayıda çalışma ve araştırma yapılmaktadır. Bu çözümlerin temelinde, öğreneceğimiz verinin modellenmesi yatmaktadır. Veriler genellikle öznitelik vektörleri ve onunla ilişkilendirilmiş sınıf etiketleri ile gösterilmektedir. Fakat bazı problemlerde bir veri birden fazla öznitelik vektörüne sahip olabilir. Böyle bir durumda modelleme işleminin birden fazla özellik vektörü ile yapılması gerekir ve bu modelleme tekniğine Çoklu Örnek Öğrenme denilmektedir.

Çoklu Örnek Öğrenme, başlangıçta ilaç molekül aktivitelerinin yapısını öğrenebilmek için ortaya atılmış olsa da, son zamanlarda metin sınıflandırma, görüntü çağrılması vb. gibi çok farklı makine öğrenmesi problemlerine de uygulanmaktadır. Temelde, bir veriye ait birden fazla öznitelik vektörleri bulunmaktadır ve her bir özellik vektörüne örnek, veriye ait bütün örneklerin birleşimine de torba denilmektedir. Çoklu Örnek Öğrenme sisteminde verilerin sınıf etiketlerinin bilgisi torbayla ilişkilendirilmiş olup örneklerin etiket bilgisi bulunmayabilir. İkili sınıflandırma problemi ele alınırsa, torba içerisindeki en az bir örnek eğer pozitif ise torbanın tamamı pozitif olarak sınıflandırılmaktadır. Eğer bütün örnekler negatif ise, torbanın sınıfı negatif olmaktadır.

Çoklu Örnek Öğrenme için önerilmiş algoritmaları üç sınıfta incelemek mümkündür. Bunlardan birincisi örnek uzayını kullanan yöntemler olarak nitelendirilmektedir. Örnek uzayında sınıflandırıcılar sadece örnekler üzerinde çalıştırılmaktadır. Bu tip yöntemlerdeki genel mantık, pozitif özellikli örneklerin negatif özellikli örnekler ile ayrışmasını sağlayarak yeni gelecek verilerdeki örneklerin sınıflandırmasıdır. Fakat örnek uzaydaki sınıflandırıcılar torbayı tamamen ele alıp öğrenme yapmadıkları için torbanın genel yapısını öğrenemezler. Ayrıca, bazı uygulamalarda torbaya ait her bir örneğin sınıf bilgisi bulunmadığından örnekler üzerinden öğrenme işlemi yapılamamaktadır. Bu problemleri ortadan kaldırmak için torba uzayında sınıflandırıcı öğrenen yöntemler önerilmiştir. Torba uzayında çalışan sınıflandırıcılar her bir örneği teker teker değerlendirmeyip torbayı bir bütün olarak değerlendirmektir. Bu tekniklerde genelde dikkat edilecek husus, torba verisinin vektörel olmayan yapısıdır. Bu şekildeki yapıların sınıflandırıcı tarafından öğrenilmesi için iki torbayı karşılaştıran bir fonksiyon gereklidir. Bu fonksiyon dolaylı olarak iki torbayı karşılaştırıp sonuç vermektedir. Fakat torbanın vektörel olmayan yapısı nedeniyle, geliştirilen yöntemlerin karmaşıklığı normal sınıflandırıcılara göre yüksek ve daha zaman alıcıdır. Bununla birlikte, öğrenilmeye çalışılan eniyileme fonksiyonunun konveks olmaması nedeniyle yerel minimum problemi yaşanabilmektedir. Bu problemleri aşmak için üçüncü bir teknik olarak,

gömülü uzayda öğrenen sınıflandırıcılar önerilmiştir. Bu sınıflandırıcılardaki temel mantık, veri tabanındaki her bir torbanın örnekleri bir fonksiyon yardımıyla doğrudan tek bir öznitelik vektörüne dönüştürülerek, elde edilen torba vektöründen makine öğrenmesinde kullanılan bilindik yöntemlerle öğrenmesini sağlamaktır. Gömülü uzaydaki yöntemin başarısı torbaların torba vektörüne dönüşümünde kullanılan fonksiyonun seçimine bağlıdır. Dönüşüm işlemini kolaylaştırmak ve verinin örüntüsünü ortaya çıkarmak için kullanılan yöntemlerden birisi seyrek kodlamadır.

Seyrek kodlama, sinyal işleme ve görüntü işleme toplulukları tarafından son zamanlarda çokça kullanılan bir yöntemdir. Bunun nedeni, seyrek kodlanacak verinin, sözlük denilen ve veriyi temsil eden temel matrisin içerisindeki temel vektörlerin çok azının doğrusal katışımıyla elde edilebilmesinden dolayıdır. Bu gösterimin faydası, veri için üst düzey bir gösterim sağlayarak verinin örüntüsünü ortaya çıkarmaktadır. Seyrek kodlama için önceden belirlenmiş temel matrisler kullanılmakla birlikte son zamanlarda yapılan çalışmalar ile temel matrisin veriden öğrenilmesinin daha iyi sonuç verdiği görülmüştür. Seyrek kodlama işlemindeki temel matrise sözlük denilmekte ve bu temel matris çıkarım işlemine ise sözlük öğrenimi denilmektedir.

İnternetin yaygınlaşmasıyla birlikte büyük verilere erişim çok kolaylaşmıştır. Fakat, büyük verilerin makine öğrenmesi tekniklerinde kullanılabilmesi için her bir verinin sınıf etiketinin belirlenmesi gereklidir. Bu işlem bazı uygulamalar için çok zaman alıcı veya pahalı bir işlem olabilmektedir. Bunun yerine, etiketsiz veri tabanından akıllıca seçim yapılarak sadece seçilenlerin etiketlerinin elde edilmesi gerçeklenebilir. Literatürde bu tekniğe aktif öğrenme denilmektedir. Aktif öğrenme sisteminde, sınıflandırıcı etiketsiz verilerden kendisine en fazla bilgiyi verecek olanları seçerek sınıf bilgisinin öğrenilmesi için bir uzmana danışır. Uzmanın verdiği sınıf bilgisi ile birlikte bu veri öğrenme verisine eklenir ve sınıflandırıcı yeni veri ile birlikte güncellenir. Bu işlem önceden belirlenmiş bir yineleme sayısı kadar tekrar eder. Öğrenme verisinin en son hali ile de son sınıflandırıcı öğrenilir.

Son zamanlarda, çoklu örnek öğrenme ve aktif öğrenme sistemleri birleştirilerek çoklu örnekli aktif öğrenme metotları geliştirilmiştir. Bu metotlar çoklu öğrenmenin hangi uzayda yapıldığına göre değişim göstermektedir. Bu yüzden çoklu örnekli aktif öğrenme metotları torba uzayında ve örnek uzayında yapılabilmektedir. Fakat bildiğimiz kadarıyla gömülü uzayda çalışan çoklu örnekli aktif öğrenme metodu üzerine çalışma yapılmamıştır. Bu tezde, seyrek kodlama ve sınıflandırıcı topluluğu tekniklerini kullanan çoklu örnekli aktif öğrenme metodu, DEMIAL algoritması önerilmiştir.

Önerilen DEMIAL algoritması genel olarak 5 farklı adımdan oluşmaktadır. Öncelikle, öğrenme verisi kullanılarak birbirlerinden farklı boyutlardan oluşan farklı sözlükler öğrenilmektedir. Bunun amacı, farklı boyutlarda öğrenilen sözlükler ile sınıflandırıcı topluluğu oluşturulup, topluluk içerisindeki çeşitliliği sağlamaktır. Daha sonra elde edilen sözlükler ile seyrek kodlama yapılarak torba içerisindeki örneklerin seyrek gösterimleri elde edilir. Seyrek kodlama örneklere üst düzeyde bir gösterim sağladığı için kullanılmıştır. Daha sonra torbaya ait örneklerin seyrek gösterimleri, birleştirilerek her bir torbanın öznitelik vektörü oluşturulur. Kullanılan birleştirme fonksiyonu örneklerin seyrek gösterimlerini özetleyen bir fonksiyon olup, örneklerin tamamını temsil eder. Öğrenme aşamasında ise torba özellik vektörleri kullanılarak makine öğrenmesinde kullanılan sınıflandırıcılardan birisi kullanılır. Bu aşamaya kadar yapılan işlemler Eğiticili Çoklu Örnek Öğrenme adımlarıdır.

Önerilen DEMIAL algoritmasının aktif öğrenme aşamasında, etiketsiz verilerin seyrek kodlaması yapılır ve her bir veri tek bir torba vektörüne dönüştürülür. Daha sonra etiketsiz veri üzerinde torba vektörü kullanılarak sınıflandırıcılardan sınıf bilgisi öğrenilir. Sınıflandırıcı topluluğunun kararı ile en yararlı veri seçilir ve uzmana sorularak sınıf bilgisi alınıp öğrenme verisine eklenir. Bu işlem belirli bir tekrarlama ile devam eder ve en sonunda son kez öğrenme verisinden sınıflandırıcı öğrenilir.

Deneysel sonuçlar 5 farklı Çoklu Örnek Öğrenme problemi içeren görüntü sınıflandırma veri kümelerinde elde edilmiştir. Öncelikle Eğiticili Çoklu Örnek Öğrenme algoritması olan SCCE-MIL yöntemi, farklı boyutlardaki sınıflandırıcı toplulukları ve farklı baz sınıflandırıcılar kullanılarak incelenmiştir. Daha sonra önerilen DEMIAL algoritması çekirdek tabanlı çoklu örnekli aktif öğrenme algoritmasıyla farklı aktif öğrenme sorgu yöntemleri ile karşılaştırılmıştır. Ayrıca baz sınıflandırıcıların DEMIAL algoritmasının başarımına olan etkileri de incelenmiştir. Deneysel sonuçlarda önerilen DEMIAL algoritmasının daha iyi sınıflandırma başarımına sahip olduğu görülmüştür.

1. INTRODUCTION

Machine Learning is a framework that models a given task by utilizing the past experience or the example data. Learning problems often comes where we cannot explicitly write a computer program about the given task. This process is an operation of optimizing a hypothesis for the given task which is called learning. After the learning, this hypothesis is used for prediction on new instances.

Machine learning techniques are mainly separated into 3 categories. First category is called supervised learning in which, the training data is given with category information namely label. Using these labels, we generalize the class information by a model to be able to identify the new example. This method is also referred as classification.

The second category in the machine learning is called unsupervised learning. In unsupervised learning, the training data has no label information and the aim is to find any pattern or information from how data is composed. Finding any pattern in data may be a challenging task than the supervised learning but it is essential for some applications where we can't receive any label information.

In many domains, obtaining data is cheap but labeling them can be laborious. Therefore, one can obtain few labeled data and huge amount of unlabeled data samples. These types of applications are named as semi-supervised learning which is the combination of the supervised and unsupervised learning. Essentially, training is done by both labeled and unlabeled data. In general, learning with unlabeled data incorporated with some labeled examples improves the learning accuracy. This is also helpful for some applications where obtaining the label information is difficult and expensive.

Essentially, the above categories only deal with data samples which have one input feature vector. But there are many applications where the data has multiple features that represent a given data. Learning process of those data is called Multiple-Instance Learning (MIL) which is an extended version of supervised learning. In MIL

framework, each data sample consists of bag - which is a collection of feature vectors called instances - and class labels are associated with bags. In some applications MIL offers a better representation of a real-world problem more than the standard supervised learning. For example, in image categorization, a single image is represented as a single feature vector in supervised learning. However, in the MIL settings, we can represent a single image with set of feature vectors which are extracted from separated regions of the image.

Nowadays with the advance of the technology, it becomes easy to obtain a vast amount of unlabeled data. In order to use this abundant data in a supervised learning framework, each data sample has to be labeled manually which can be an overwhelming task to accomplish. Instead of labeling all the unlabeled data, one can "smartly" select some of them using an active learning framework. Active learning is a well-known and widely used framework for many machine learning problems where obtaining data labels are difficult and expensive. In this framework, active learner is allowed to choose the most informative unlabeled data and ask (query) its label from the oracle to perform better with less training data.

In order to combine the MIL framework and the active learning framework, we need to represent the instances in the bags more efficiently so that the classifiers have better results. This can be achieved with sparse coding techniques. Sparse coding is a well-known technique in signal processing and image processing communities which allow us to represent signals with the linear combination of the basis vectors called atom and the collection of the atoms are called dictionary. This technique gives us a general representation of a signal in terms of sparse combinations of atom vectors. Also, sparse coding removes the possible noise from the signal.

In this thesis, we propose an MI active learning method, namely DEMIAL (Dictionary Ensemble based Multiple Instance Active Learning), which employs classifier ensembles that is trained in sparse represented data. We compare our method with different ensemble techniques, ensemble combination methods.

1.1 Contribution of the Thesis

This thesis composed of four different aspects which bring novelty to the MIL framework:

- We propose an MI active learning approach using classifier ensemble system which is trained with a sparse representation of the data from the multiple different sized dictionaries.

- We analyze the active learning query strategies for the proposed algorithm and compare these with different approaches in the MIL framework.

- Ensemble system is constructed by using different sized dictionaries in which ensemble size is investigated.

- DEMIAL algorithm is investigated with different base classifiers and active learning query strategies for image categorization.

1.2 Thesis Structure

The remaining chapters of the thesis are as follows. In Chapter 2, we cover the methods and techniques that are used in this thesis as background information. In Chapter 3, we will discuss the MIL framework with sparse coding and classifier ensemble methods. In Chapter 4, we will present the, DEMIAL, MI active learning method with sparse coding and classifier ensemble. In Chapter 5, we will conclude the thesis by giving the take home messages.

2. BACKGROUND

In this chapter, we will cover some of the previous works about MIL and other methods that are used in this thesis.

2.1 Multiple-Instance Learning

Multiple-Instance Learning is a machine learning framework proposed by Dietterich et al [1]. In MIL framework, each sample in a dataset is a set of feature vectors representing a sub-part of the sample itself called instances. If we rephrase the last statement in MIL terms, each data sample in the dataset is a bag of instances. Each bag has a class information in the dataset and instances may have a class association but may not be known. In binary classification, a bag is considered positive if at least one of the instances is positive. If all the instances are negative, then the bag label is negative. The MIL framework is represented in Figure 2.1.

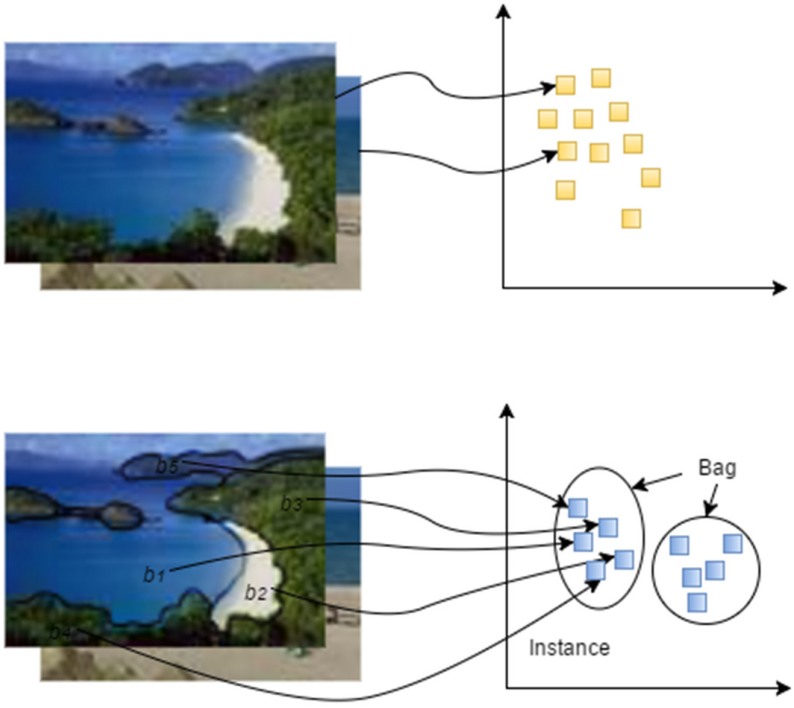

Figure 2.1 : The difference between single instance learning and multiple instance learning.

Mathematically, each bag in the dataset can be shown as $B_i = \{b_{i1}, b_{i2}, \dots, b_{iK_i}\}$ where b_{ij} is the *jth* instance vector of the *ith* bag. Class information is represented as y_i and it is associated with the B_i. Note that each bag has a different number of instances which is denoted as K_i. For binary classification, y_i becomes positive if at least one instance in B_i is positive. If all the instances in the bag are negative, then y_i considered as negative.

To solve MIL problems, there are several approaches in the literature. Mainly, MIL approaches are separated in three categories: Instance-Space (IS), Bag-Space (BS) and Embedded Space (ES) [2]. We will investigate these categories more detail in the following sub-sections.

2.1.1 Instance-space algorithms

Algorithms that use only instances in the bags are considered instance-space (IS) algorithms. In the IS algorithms, label information is considered as related to instances and classifiers are trained by positive instances in the bags.

Multiple-Instance Learning was firstly proposed by Dietterich et al. [1] for the prediction of drug molecule activity. In their work, they proposed the first MIL method named Axis-Parallel Rectangle (APR) algorithm which builds a hyper-rectangle around the positive instances of each bag while not including the negative instances.

After that, Maron et al [3] proposed the *diverse density* (DD) algorithm for MIL framework. Unlike the APR algorithm which encloses the instances with a hyper-rectangle in the instance-space, DD algorithm tries to find a single vector in the instance space that has minimum distance with the positive instances while has maximum distance with the negative instances. Mathematically, the classifier is a vector $h = \{w\}, w \in \mathbb{R}^d$ and the posterior probability of the instance b_{ij} given the features depends on the distance between it and the classifier vector:

$$p_{ij} = \exp\left\{-\left\|b_{ij} - w\right\|^2\right\} \tag{2.1}$$

Specifically, they defined a positive region with a Gaussian centered vector w with this probability. To calculate this region, they defined an optimization function with all of the positive instances fall in the region and the negative ones are out of it. This

optimization is done with a noisy-OR model for calculating the likelihood of the data sample and the gradient descent for training the optimal classifier vector [4].

Following the same principle, Zhang and Goldman [5] has proposed EM-DD algorithm which uses Expectation–Maximization (EM) algorithm for the DD optimization. The EM-DD, algorithm first estimates the most probable instances in the bags to have the positive label. Then according to its predictions, it finds the classifier vector which has the maximum distance between the negative instances and minimum distance between the positive instances. This process will continue until no further improvement is achieved or a predefined set of iterations.

Alternatively, Andrews et al [6] proposed mi-SVM algorithm which uses only instance information. In their algorithm, the goal is to find an MI-separating hyperplane which separates at least one positive instance and thus creates halfspace of positive and negative instances.

2.1.2 Bag-space algorithms

In instance-space, algorithms train a model to estimate instances' labels and discriminate the positive instances against the negative instances. Thus, this type of algorithms train on local information. In contrast, bag-space algorithms train a model using the bags and this allows the algorithm to capitalize more information especially bag information.

Since the bags are non-vector entity, we have to define a function which compares two bags in the dataset. Usually, comparison function calculates the distance of the two bags B_i and B_k. In the literature there are several different distance functions that are used in MIL settings. One of them is the *minimal Hausdorff distance* [7]:

$$HD(B_i, B_k) = \min_{b_{ij} \in B_i, b_{kl} \in B_k} \left\| b_{ij} - b_{kl} \right\| \tag{2.2}$$

This function gives the minimum distance between two bags' instances. The other distance function is *Earth Movers Distance* (EMD) [8] :

$$EMD(B_i, B_k) = \frac{\sum_j \sum_l w_{jl} \left\| b_{ij} - b_{kl} \right\|}{\sum_j \sum_l w_{jl}} \tag{2.3}$$

where w_{jl} are the weights of the instances that are calculated by an optimization process which minimizes the $EMD(B_i, B_k)$ with some constraints.

Aside from the distance functions, *kernel functions* are also used as a similarity measure. Thus one can benefit from the advantages of kernel functions. Gartner et al. [9] has proposed set kernel function:

$$K_{set}(B_i, B_k) = \sum_{b_{ij} \in B_i, b_{kl} \in B_k} k(b_{ij}, b_{kl}) \tag{2.4}$$

where $k(.,.)$ is any kernel function e.g. Gaussian, polynomial, linear, etc. that uses instances b_{ij} and b_{kl} as the input. To overcome the varying cardinality of the instances the set kernel similarity is normalized using the following formula:

$$K_{nset}(B_i, B_k) = \frac{K_{set}(B_i, B_k)}{\sqrt{K_{set}(B_i, B_i)}\sqrt{K_{set}(B_k, B_k)}} \tag{2.5}$$

2.1.3 Embedded-space algorithms

Embedded-space algorithms are similar to the bag-space algorithms. In Bag-space, the bag features are constructed by using distance or kernel functions implicitly. In Embedded-space algorithms, bags are converted into a single feature vector explicitly by using a mapping function.

In that sense, Gartner et al [9] proposed a *statistic kernel* which aggregates the instances in a statistical approach. The intuition of the statistic kernel (which is also called *pooling function* in some other areas) is to summarize the bag into a single representative vector by using statistics e.g. mean, median, min-max etc. This technique allows us to use different statistics about the data. As an example to the pooling function, max pooling function can be given as follows:

$$\bar{B}_i(k) = \max\{|b_{i1}(k)|, |b_{i2}(k)|, \dots, |b_{ij}(k)|, \dots\} \tag{2.6}$$

where $\bar{B}_i(k)$ is the *kth* element of the *ith* bag feature and $b_{ij}(k)$ is the *kth* element of instance b_{ij}.

Alternatively, Chen and Wang [10] proposed DD-SVM method that combines the DD method and Support Vector Machines (SVM). This algorithm has two steps: 1)

calculate the instance prototypes using the DD algorithm, 2) each bag is mapped into a new feature space using the learned instance prototypes. Then they trained a SVM using these new features. Similar to DD-SVM, Chen et al. [11] proposed MILES algorithm that extends the DD method. Their approach is to calculate the instances that are close to classes by embedding the instances and selecting the features that are most relevant.

2.2 Classifier Ensembles

In the real world, we often find ourselves with a decision which we can't decide it by ourselves. At that time, we usually obtain other people's opinion and combine their decisions with our's. This natural process can be applied to the machine learning framework in which training multiple classifiers and combining the decisions of them is called classifier ensemble system. By combining classifiers, our aim is to achieve more accurate classifier system than a single classifier which reduces the possibility of a wrong classification [12].

Another aspect of a classifier ensemble is the usability with too large size of instance space. In some applications, large amount of the data has to be learned for an effective classifier but data size makes the learning process not practical to be handled by a single classifier. This problem can be solved by constructing a classifier ensemble in which, each classifier is trained with the subset of the whole dataset. This solution is proved to be much more efficient approach rather than the single classifier trained on the whole dataset [13].

Similarly, to the size of the dataset, there are some applications in which each data sample has large feature vector. These kinds of applications become troublesome for single classifiers because complexity of the learning process is getting higher with more features which is called curse of dimensionality problem. To solve this, an ensemble system can be constructed with classifiers learned with the subspaces of the feature vector. This technique allows us to select some features in the dataset and discards the others which reduce the complexity of the learning algorithm.

Another problem with a single classifier is that the data sample can be too complex for a single classifier to solve by itself. Specifically, for a given dataset, decision boundary of the data can be too complex for the classifier to fit. These complex

boundaries can be approximated by dividing the data into sections and training a classifier for each section and combining the results. This divide-and-conquer-like strategy is appropriate for much more complicated problems.

Ensemble system is useful for many applications. But this requires classifiers to be diverse in their decisions to boost the overall classification. To be more specific, each classifier should make different errors so that combination of the classifiers reduces the total error. This means that each of the classifier doesn't have to be the best to classify the data by itself but the combination with the other classifiers achieve better results [12].

All classifier ensemble systems have two important functions that we build ensemble system upon. One of them is creating a set of classifiers that are diverse enough and the other one is the combination of the results. We will discuss these functions in following sub-sections.

2.2.1 Creating the classifier ensemble

In order to create an ensemble system, we have to find a way to generate multiple base classifiers which are different enough from each other. There are several methods in the literature for creating diverse ensemble system.

One of the early and most used approach for ensemble system is called *bootstrap aggregating* (or in short *bagging*) in which, different data subsets are randomly selected from the whole training dataset. Each subset of the data is used for training a single classifier. But using random selection technique would yield duplicate data on multiple subsets of the dataset with probability of approximately 0.368. To create diversity in bagging, unstable base classifiers, in which small changes in the dataset would yield larger output changes e.g. decision tree and multi-layer perceptron, is recommended [13].

Alternative to the bagging, boosting is proposed by Schapire [14] which generates weak classifiers but the combinations of them make strong learner. Boosting also uses resampling of the data but selects with a strategy. The classifiers that use subset of data are created iteratively. At first, classifier is trained by randomly selected subset data. For the next classifier, the most informative data is selected given the first classifier and trained accordingly. The third classifier's data is selected which first and second classifier disagree [12].

As a more general approach, AdaBoost is proposed by Freund and Schapire [15]. In AdaBoost, for each classifier in the ensemble, data samples are selected with an iteratively updated distribution. In this distribution, the likelihood of the given data sample is increased when it is misclassified. This ensures that in the next iteration misclassified data is used most probably for the training.

Another approach for diverse ensemble is to adjust the training parameters of the classifier so that each classifier's decision boundary differs from each other. Training parameters control the instability of the classifier and thus the diversity can be achieved [12]. Lastly, diverse ensembles can be constructed by training classifiers that uses different feature subsets of the data. This method is called random subspace method [16].

2.2.2 Combining the ensemble results

The second most important function for ensemble system is combining the results of the multiple classifiers for a given data. The result of the ensemble depends on the decisions given by the classifiers. Since we assume that classifiers' results are different enough from each other, combination strategy is crucial for ensemble system. In general, there are two group of combination strategies: combination of class labels and combination of class posterior probabilities.

There are several methods for combining the class labels. The straightforward method would be to classify the data with the label on which majority of the classifiers agree. This method is called *majority voting* and can be described as,

$$\sum_{i=1}^{L} d_{i,k} = \max_{j=1\dots C} \sum_{i=1}^{L} d_{i,j} \tag{2.7}$$

where $d_{i,j}$ is the binary decision of the *ith* classifier about the class *j*, L is the ensemble size and k is the chosen class. Note that majority voting can be used under some assumptions: 1) L has to be odd, 2) probability of each classifier i to give the correct class label is equal for any data and 3) classifier outputs are independent of each other [13].

If we have a prior knowledge that some classifiers are somewhat more competent than other classifiers, we can weight their outputs and combine them. This technique

is called weighted majority voting and can be shown as,

$$\sum_{i=1}^{L} w_i d_{i,k} = \max_{j=1\ldots C} \sum_{i=1}^{L} w_i d_{i,j} \tag{2.8}$$

where w_i is the weight of the *ith* classifier. For this technique, the most important problem is to assess the classifiers quality and weight their contribution. If we have the prior knowledge of the classifiers performance, it won't be a problem. But if we don't have, we have to come up with a strategy. In some ensemble techniques like AdaBoost, the assessment is made while the ensemble is constructing. For others, we can split a validation dataset and assess the classifiers performance on it.

Beside class labels, classifiers may have a continuous output that shows the likelihood of a given class. For the combination of the class likelihoods, there are several methods in the literature but the most common used ones are called algebraic combinations. The outputs are combined in an algebraic function that supports the best results. These functions are represented in Table 2.1.

Table 2.1 : Algebraic ensemble combiners.

Name	Formula
Product Rule	$\mu_k(x) = \frac{1}{L}\prod_{i=1}^{L} d_{i,k}(x)$
Mean Rule	$\mu_k(x) = \frac{1}{L}\sum_{i=1}^{L} d_{i,k}(x)$
Max Rule	$\mu_k(x) = \max_{i=1\ldots L}\{d_{i,k}(x)\}$
Median Rule	$\mu_k(x) = \underset{i=1\ldots L}{\text{median}}\{d_{i,k}(x)\}$
Minimum Rule	$\mu_k(x) = \min_{i=1\ldots L}\{d_{i,k}(x)\}$
Weighted Average	$\mu_k(x) = \frac{1}{L}\sum_{i=1}^{L} w_i d_{i,k}(x)$

Each function leverages some quality of the outputs and some functions are more suitable for some ensemble techniques. For example, for AdaBoost, weighted average is more suitable since the algorithm calculates the weights of the classifiers.

The best combination method depends on the problem and our prior knowledge of the problem [12].

2.3 Sparse Coding

In signal processing, sparse coding is a well-known technique to approximate a given signal by combining few redundant basis. The advantage of the sparse coding is that signals can be well approximated and the resulting sparse features capture the structures or patterns of the signals [17]. In the following sub-sections, we will try to cover the methods and techniques of sparse coding. We note that sparse coding will be used in MIL framework so, we concatenate all the training bags' instances and form instance matrix $\tilde{b} = \{b_1, \dots, b_M\}$ where $M = \sum_i^{N\{T\}} K_i$. These notations are used for this subsection.

2.3.1 Definition

Essentially, sparse coding is a problem of a linear system $D\alpha = \tilde{b}$ that have some conditions and constraints. In this equation, $D \in \mathbb{R}^{n \times m}$ is a full-rank matrix with $n < m$ which makes the matrix D over-complete. This results the linear system to have infinite solutions. But we are interested in sparse solutions which are the least non-zero elements in $\alpha \in \mathbb{R}^m$.To enforce the sparsity in a linear equation, we need a constraint in the equation. In particular, we need to narrow down the infinite number of solutions to fit our needs. This can be done by regularization,

$$\min_{\alpha} J(\alpha) \quad s.t. \ \tilde{b} = D\alpha \tag{2.9}$$

where $J(\alpha)$ is any regularization function. The most common choice of this function is ℓ_2 norm which is widely used for many engineering problems for being simple and provide unique solution. Although it gives a unique solution, it is not the sparsest solution [18].

Since we are trying to find the sparsest solution, the intuitive regularization would be the ℓ_0 norm:

$$\min_{\alpha} \|\alpha\|_0 \quad s.t. \quad \tilde{b} = D\alpha \tag{2.10}$$

The ℓ_0 norm is actually not a norm because it doesn't meet the requirements of a norm, but as a concept it shows the best notion of the sparsity. ℓ_0 pseudo-norm gives the number of non-zero elements in the vector. If we try to solve this optimization problem, we cannot do it easily. Because, ℓ_0 is not a convex function so standard convex methods don't apply to it. Moreover, even if we find a solution, we cannot guarantee its uniqueness and validity of the optimal solution [18]. In short, solving ℓ_0 pseudo-norm is a NP-Hard problem. In general, there are two types of approaches for this challenging problem.

2.3.2 Greedy approaches

Originally, our optimization problem is given in equation 2.10. But if we consider the real world problems, generally we will have noise in the dataset. So our optimization problem turns into:

$$\min_{\alpha} \left\| \tilde{b} - D\alpha \right\|_2^2 \quad s.t. \quad \|\alpha\|_0 \leq k \tag{2.11}$$

or

$$\min_{\alpha} \|\alpha\|_0 \quad s.t. \quad \left\| \tilde{b} - D\alpha \right\|_2^2 \leq \varepsilon \tag{2.12}$$

where we state the basis matrix D as dictionary and α as sparse representation of the signal $\tilde{b}$. As we mentioned in previous section, solving this optimization is an NP-Hard problem. Since we cannot solve by evaluating all the combinations which is not practical, we can choose greedily so that at each step we select the best atom in the dictionary. This technique is called Matching Pursuit (MP) which is proposed by Mallat and Zhang [19]. This algorithm iteratively selects the closest atom with the signal. After the selection, the signal is updated so that the contribution of the atom is removed and those steps continue with the residual signal until it convergence.

The issue with the MP algorithm is that it may have large number of iteration for convergence. Because in optimization step, it may select the same atom from the previous steps. To overcome this problem, Pati et al. [20] proposed Orthogonal Matching Pursuit (OMP) algorithm. In OMP algorithm, it sweeps all the atoms in dictionary - except the previously selected ones - and selects the closest atom to the residual signal. Then, it minimizes the residual signal with the selected atoms and

update their contribution in the sparse vector. Not using the selected atoms and the sparse vector update step guarantees the orthogonality between the residual signal and the previously selected atoms [18].

2.3.3 Relaxation techniques

In the previous section, we try to minimize the equation 2.11 and 2.12 but these optimization problems are non-convex functions. Another approach for solving the optimization for sparse coding is relaxing the ℓ_0 pseudo-norm to ℓ_1 norm. This will convert the optimization problem into:

$$\min_{\alpha} \|\alpha\|_1 \quad s.t. \quad \left\|\tilde{b} - D\alpha\right\|_2^2 \leq \varepsilon \tag{2.13}$$

Converting the actual optimization problem may not give the sparsest solution as in ℓ_2 norm but Donoho [21] has shown that the ℓ_1 norm minimizations are equal to ℓ_0 pseudo-norm minimizations if the solution is sparse enough.

The early research is done by Chen et al [22] and they proposed Basis Pursuit for solving sparse coding problem. In basis pursuit, the optimization problem is converted into a linear programming problem and solved in that sense. Alternative to basis pursuit, Efron et al [23] proposed Least Angle Regression (LARS) algorithm which is similar to OMP algorithm. LARS algorithm is an iterative algorithm similar to OMP but it solves the relaxed optimization problem. At each step, it selects the best atom from the dictionary which is the closest to the signal. The difference between the OMP algorithm is that it guarantees the global optimizer of the equation 2.13 [18].

2.4 Dictionary Learning

In the previous section, some sparse approximation techniques are presented. These techniques are established on a dictionary. In particular, it is essential to choose a dictionary for calculating the sparse representations. There are many pre-constructed dictionaries for sparse coding e.g. wavelets, curvelets, Gabor dictionary etc. These fixed dictionaries achieve relatively fast results and usually have theoretical analysis about how the sparsity of the signal is achieved but they are used for a specific problem/signal type. This problem leads to algorithms that learn dictionary from the

data itself. The aim of the dictionary learning is to learn the most suitable dictionary from the data rather than the mathematical descriptions of the data [18]. This aspect allows us to adapt the dictionary to the data without changing the type of the algorithm.

Mathematically, given a set of signals $\tilde{b}$, we want to learn a dictionary D such that each signal can be approximated as a linear combination of few atoms. This can be done by solving one of the following dual optimization:

$$\min_{D,\{\alpha_i\}_{i=1\dots M}} \sum_{i=1}^{M} \|\alpha_i\|_1 \quad s.t. \quad \left\|\tilde{b}_i - D\alpha_i\right\|_2^2 \leq \varepsilon, \quad 1 \leq i \leq M \tag{2.14}$$

or

$$\min_{D,\{\alpha_i\}_{i=1\dots M}} \sum_{i=1}^{M} \left\|\tilde{b}_i - D\alpha_i\right\|_2^2 \quad s.t. \quad \|\alpha_i\|_1 \leq k, \quad 1 \leq i \leq M \tag{2.15}$$

Note that those two optimization problems pose different optimization criteria but both of them enforce the sparsity of the signal. These optimization problems can be solved in two steps: sparse coding step and dictionary update step.

First algorithm that learns dictionary is proposed by Engan et al. [24] and they called Method of Optimal Directions (MOD) algorithm. MOD is an iterative optimization algorithm which solves the equation 2.14. At iteration *t*, sparse coding is done using the *(t-1)th* dictionary $D_{(t-1)}$ with any pursuit algorithm. To calculate the new dictionary $D_{(t)}$, we use sparse representations of the data $\alpha_{(t)}$ by

$$\begin{aligned} D_{(t)} &= \underset{D}{\operatorname{argmin}} \left\|\tilde{b} - D\alpha_{(t)}\right\|_F^2 \\ &= \tilde{b}\alpha_{(t)}^T \left(\alpha_{(t)}\alpha_{(t)}^T\right)^{-1} \\ &= \tilde{b}\alpha_{(i)}^+ \end{aligned} \tag{2.16}$$

where $\|\,.\,\|_F^2$ is the Frobenius norm. This process is continued until the convergence.

Another dictionary learning method is proposed by Aharon et al. [25] called K-SVD method. This method is similar to the MOD method with a different dictionary update step. In K-SVD, each atom in the dictionary are updated separately so that, equation 2.15 is converted into

$$\left\|\tilde{b} - D\alpha\right\|_F^2 = \left\|\left(\tilde{b} - \sum_{j \neq j_0} d_j \alpha_j^T\right) - d_{j_0}\alpha_{j_0}^T\right\|_F^2 \tag{2.17}$$

where d_j is the atom of the dictionary *D*. For optimal d_{j_0}, the contributions of the other atoms are extracted from the residual signal E_{j_0} which is

$$E_{j_0} = \tilde{b} - \sum_{j \neq j_0} d_j \alpha_j^T \tag{2.18}$$

After the extraction, SVD is applied to remaining E_{j_0} and largest eigenvector is assigned to d_{j_0} as the new value. This process is performed for all atoms thus named as K-SVD.

Both MOD and K-SVD algorithms iterate through the training batch data which means at each dictionary optimization step all the training data is used. But if the training data is changing over time e.g. video sequences, those algorithms can't work. To overcome this challenge, Marial and Bach [26] proposed Online Dictionary Learning method which uses block coordinate descent algorithm with the help of the previously learned dictionary and sparse representations of the signals.

3. DICTIONARY ENSEMBLE BASED MULTIPLE INSTANCE LEARNING

In this chapter, we will cover the dictionary ensemble based MIL algorithm. This algorithm is the basis of the proposed method which will be covered in the next chapter.

3.1 Dictionary Ensemble Based MIL Algorithm

Recently, Song et al [27] proposed Sparse Coding and Classifier Ensemble based MIL method (SCCE-MIL) for image categorization. Their algorithm combines the ensemble strategy with sparse coding for the MIL framework. Overview of the SCCE-MIL algorithm can be shown in Figure 3.1. Essentially, their algorithm has 4 steps. Firstly, ensemble system is constructed by learning different sized dictionaries with the training data. Since dictionaries are trained with different sizes, they capture different structures of the signal. This aspect would yield diverse ensemble system that boost the overall classifier accuracy.

In order to train a dictionary in a MIL setting, we have to concatenate all the instances in the bags into an instance matrix. So, the resulting dictionary modals the instances of the bags. After the dictionary training, each instance in the bags is transformed into multiple sparse feature vectors using different sized dictionaries. While these sparse features represent the key elements in their own signal, they give higher level information about the data sample i.e. patterns in the signal. Also using sparse coding techniques removes the noises from the signal so that we can use sparse features instead of original features [28].

As we reviewed in Chapter 2.1, there are three approaches to solve a MIL problem. In SCCE-MIL algorithm, sparse representations for each instance is obtained. Using these representations, a single bag representation using max-pooling function is computed. Max pooling function allows us to convert the bag instances into a bag feature so that it summarizes the most influential features in the bag. Since the instances in the bags are sparse, calculating the maximum value would give best

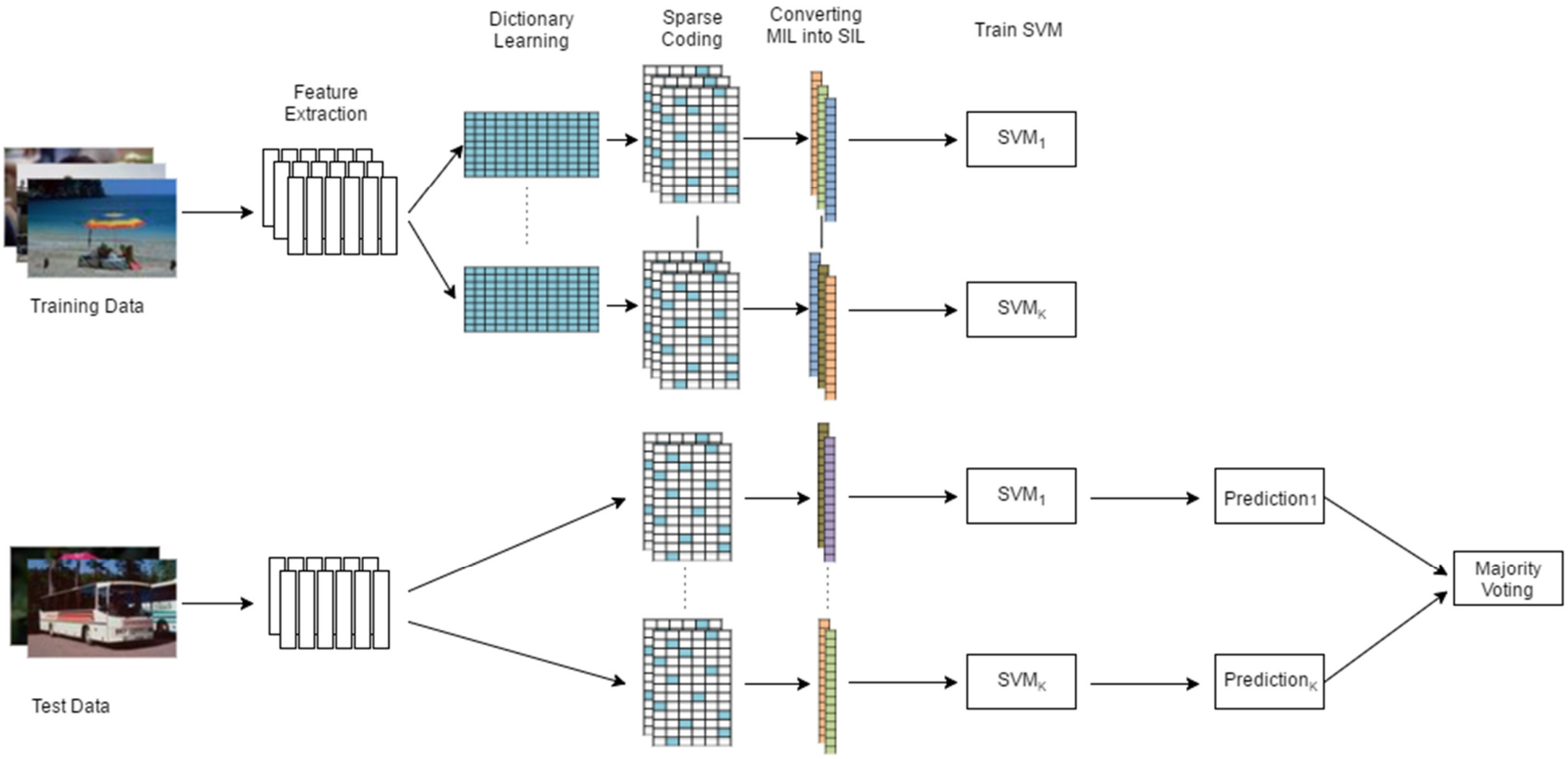

Figure 3.1 : Overview of SCCE-MIL algorithm.

contribution of the atom in the dictionary. This aspect of the max proven to be true for applications e.g. image categorization [29].

As the last step of SCCE-MIL algorithm, Support Vector Machines (SVM) is trained to learn the bag representations. Since SCCE-MIL is an ensemble algorithm multiple SVMs are trained on different sized bag features which are extracted from different sized dictionaries. Pseudo-code of the algorithm is given in Table 3.1.

Table 3.1 : SCCE-MIL algorithm.

Input: Training set $B^{\{T\}}$, dictionary sizes $S = \{s_1, \dots, s_L\}$, ensemble size L Output: Trained set of SVM's
1. Concatenate all the instances in the bags and construct the instance matrix $\tilde{b} = \{b_1, \dots, b_M\}$ where $M = \sum_i^{N^{\{T\}}} K_i$.
2. for i=1:L
3. Train an $S(i)$ sized dictionary.
4. for j=1:size($B^{\{T\}}$)
5. Calculate the sparse representations of every instance in the *jth* bag
6. Calculate the bag feature $\bar{B}_j$ using the *max pooling function*
7. end
8. Train the SVM_i classifier using $\bar{B} = \{\bar{B}_1, \dots, \bar{B}_j, \dots\}$
9. end
10. Return ensembles of SVM_i's

After the training step, the test data is processed like the training data but, dictionary learning and SVM training steps are omitted. The trained dictionaries are used to obtain sparse coding of the test instances and trained SVM's are used for classification. Once the results of the each SVM is calculated, their class decisions are combined using majority voting and the final result would be the class label of the test data.

Song et al [27] also analyzed the effect of the dictionary size but they didn't consider the best ensemble size and the base classifier. In the next sections, experiments are carried to investigate those issues.

3.2 Experimental Setup

Experiments are conducted on Corel 1000, 2000 [11], Elephant, Fox and Tiger [6] datasets. Corel 1000 and 2000 datasets are image dataset containing 10 and 20 different categories respectively. In the dataset, each category has 100 images and each image is a 384×256 or 256×384 sized JPEG file. Images are divided into non-

overlapping blocks of size 4x4 pixels. For constructing the image features, three separate feature extraction methods are applied. The first method is to average the LUV color components of the blocks. Secondly, wavelet transform is applied and squire root of energy values in the high frequency band is used. Lastly, the shape properties are calculated for each region. Once the feature extractions are made, all of the properties are aggregated as a one feature vector. The resulting feature vector for Corel 1000 and 2000 dataset is a 9-dimensional feature vector. In our experiments we used the calculated image features from [11]. Some example images in the Corel 1000 dataset are shown in Figure 3.2.

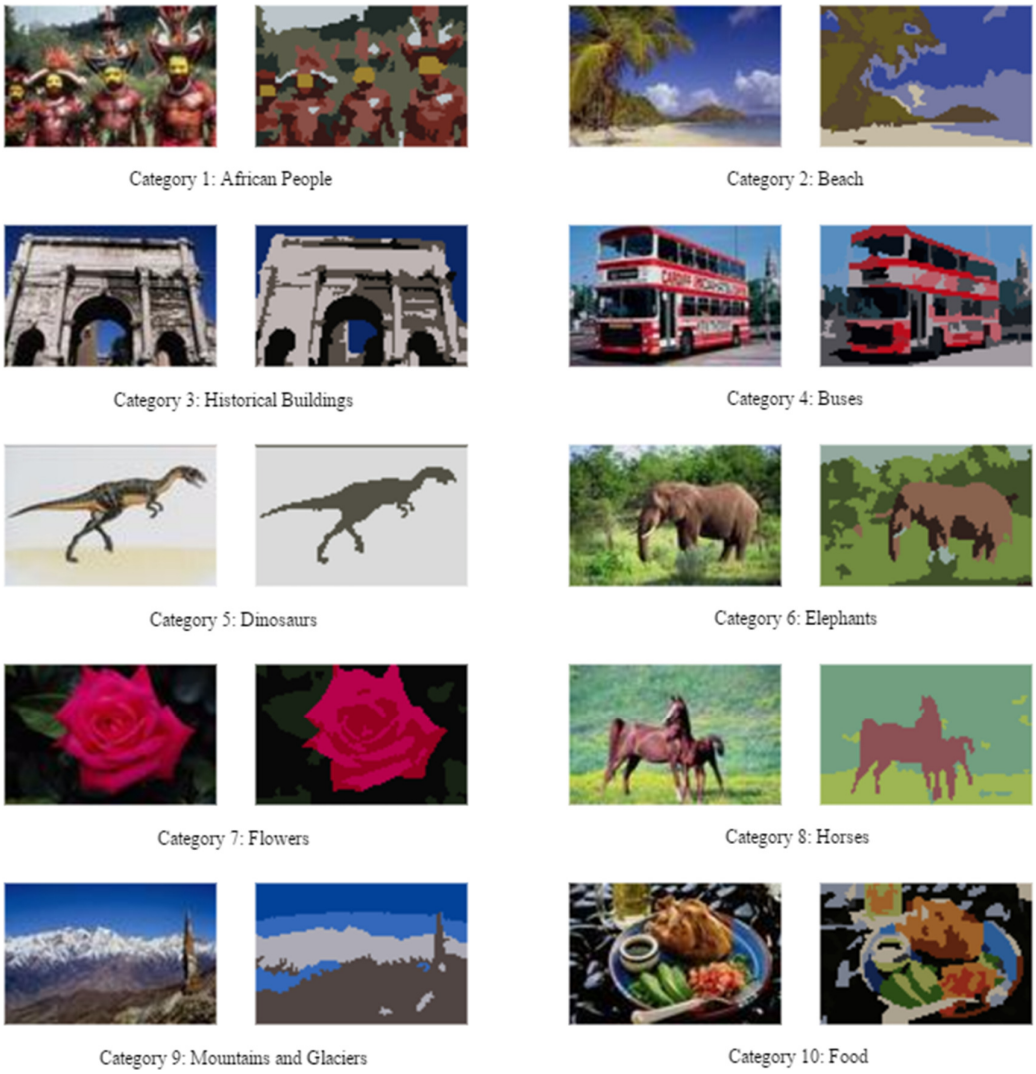

Figure 3.2 : Example images of COREL 1000 dataset [11].

Elephant, Fox and Tiger datasets are three image datasets where each of them contains 100 of animal images and 100 of other images in total of 200 images. All

images are separated into regions and each region is represented as a 320-dimensional feature vector which describes the color, texture and shape property of its region. For Elephant, Fox and Tiger datasets, we used the extracted image features from [6] directly.

For each dataset, we conducted the experiments 10-fold cross validation and average results are reported. For dictionary learning and sparse coding, SPAMS toolbox [26] is used. In SPAMS, we used LARS algorithm for sparse coding and Online Dictionary Learning for dictionary learning. In SCCE-MIL algorithm, Song et al. conducted their tests with SVM as the base classifier. We extended their research with different base classifier - Decision Tree (DT), Multi-Layer Perceptron (MLP) - and compare their performances. We also analyzed the ensemble size of the SCCE-MIL method. The implementation of the SVM is done with LIBSVM toolbox and the parameters are selected using grid search and best parameters are used. For MLP and DT, MATLAB toolbox is used.

3.3 Experimental Results

Classification accuracies with respect to the ensemble size of the SCCE-MIL algorithm are shown in Figure 3.3. As it can be seen from the figure, in Corel 1000, Corel 2000 and Elephant datasets, SVM gives the highest classification accuracy over DT and MLP. In Song et al. [27] work, they used Corel 1000 and 2000 for their experiments and their results are consistent with our experiments. But for Fox dataset, SVM performs poorly for large numbers of classifiers comparing with DT and MLP. For Tiger dataset all of the algorithms converge to the same accuracy level when the ensemble size is 40. Besides, in Fox dataset, DT performs significantly better than the other base classifiers when the ensemble size is 40. So the structure of the SCCE-MIL algorithm is powerful for image categorization but we need to test the base classifier according to the dataset.

When we increase the ensemble size of the system, classification accuracy increases for some dataset. For example, if we look for Corel 1000 and 2000 datasets, we can see the improvement of the accuracies. But, this result doesn't apply to the other datasets. For example, in Elephant and Tiger datasets, all the algorithms deviate in terms of accuracy. In Fox dataset, SVM performance drastically decreases. The best performance can be achieved by DT for Fox dataset. Note that in SCCE-MIL

algorithm, authors constructed the classifier ensembles in an arbitrarily selected size, 40. As we can see from the figures, the best accuracies are generally achieved with the ensemble size 30.

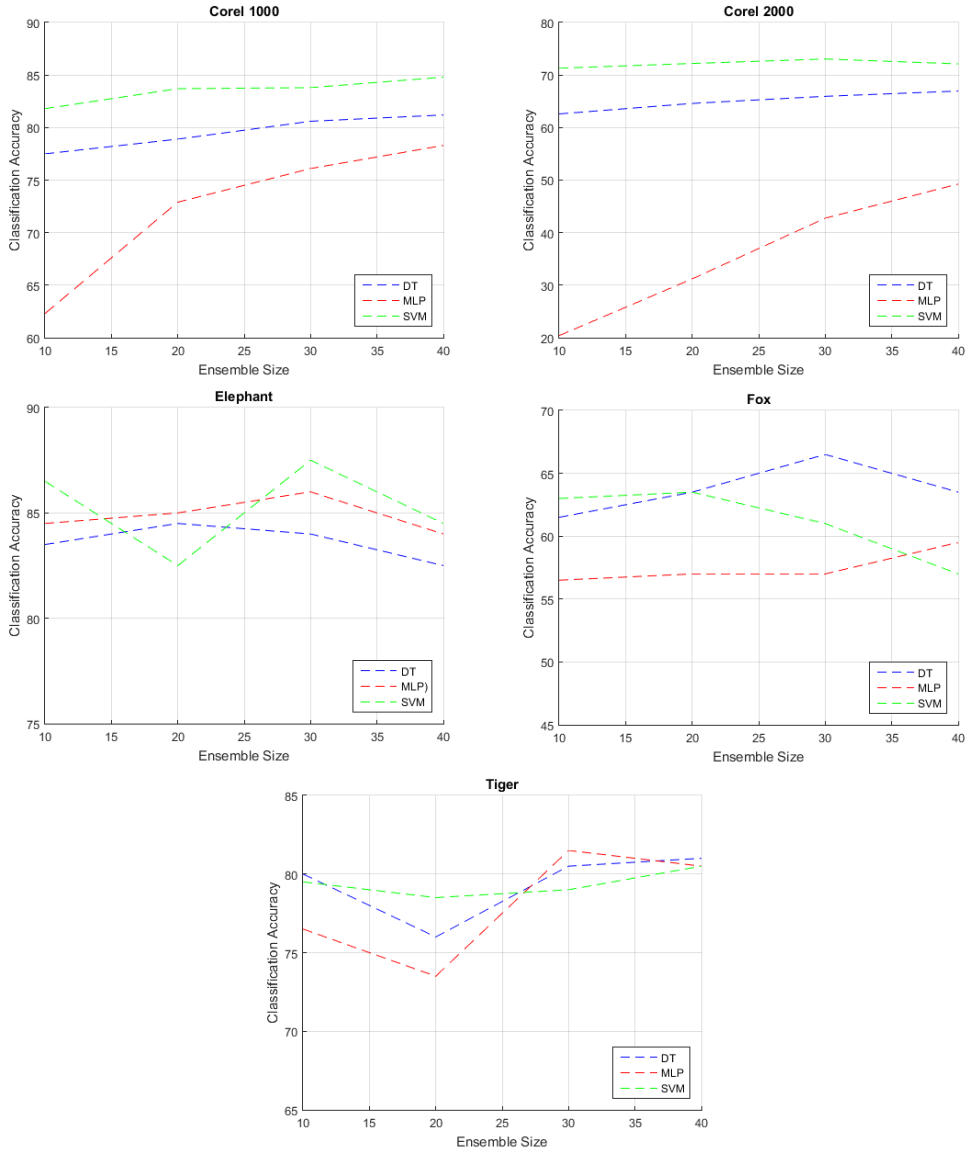

Figure 3.3 : Result of the SCCE-MIL algorithm with different ensemble size and base classifier.

With these experiments, we investigate the issues regarding to the ensemble size and the base classifier selection. Our findings show that the best classifier and the ensemble size for the SCCE-MIL algorithm is a relative issue to the database which we try to model. In order to narrow down those issues for real applications, one needs to use a validation test to try to optimize the base classification and ensemble size.

4. DICTIONARY ENSEMBLE BASED MULTIPLE INSTANCE ACTIVE LEARNING

After the explanation of the SCCE-MIL algorithm, now we propose our Dictionary Ensemble Based Multiple Instance Active Learning algorithm, namely DEMIAL. Firstly, we will explain active learning and some strategies in the literature. Later, we will discuss our approach and present the results.

4.1 Active Learning

In many real world machine learning problems, we encounter with few labeled data samples and abundant sized unlabeled data. Instead of modeling the data only with the labeled ones, one may try to include the unlabeled data. But this can be an enormous work to accomplish. Another approach would be that instead of labeling all the unlabeled data, we may choose some of the instances wisely and label them only. This technique is called active learning and it is widely used in many machine learning problems. The intuition is that active learner has the opportunity to select the unlabeled data so that learner yields better results with less data. This allows us to build classifiers with a dataset in which labeling is very expensive, difficult or time-consuming. In active learning terminology, labeling process is stated as *querying* the data label to an *oracle* (human or computer expert).

In active learning framework, the most important function is to find the most informative data sample in an unlabeled dataset. We need the informative data sample because it contributes to the classifier the most. Thus with the query function, active learning allows us to train better classifiers with fewer data samples.

Among the query strategies in the literature, the simplest and most used one is called *uncertainty sampling* [30]. In uncertainty strategy, we choose the unlabeled data which the classifier is the least certain about its label. If we think of a probabilistic classifier model for a binary classification problem, the unlabeled data sample having 0.5 posterior probability is selected and queried for its label [31]. As for non-probabilistic case, the data sample that is close to the decision boundary is queried.

Uncertainty can be measured using the following formula:

$$B^* = \underset{B_i \in B^{\{U\}}, i=1 \dots N^{\{U\}}}{\operatorname{argmax}} 1 - P_\theta(y|B_i) \tag{4.1}$$

where $P_\theta(y|B_i)$ is the posterior probability under the model θ and B^* is the queried data sample.

This strategy is quite straightforward and intuitive but it only examines the most probable class information. This is not a problem for a binary classification problem, but for multi-class problem it may mislead. In particular, it doesn't consider the other classes' probabilities. To fix this problem, Scheffer et al. [32] proposed *margin sampling*:

$$B^* = \underset{B_i \in B^{\{U\}}, i=1 \dots N^{\{U\}}}{\operatorname{argmin}} P_\theta(y_1|B_i) - P_\theta(y_2|B_i) \tag{4.2}$$

where y_1 and y_2 is the most probable two classes under the model θ. As we can see, margin sampling selects the minimum margin between class posterior probabilities. Querying the small margin gives the most ambiguous data sample between the two classes. But if we need the most ambiguous data sample among the dataset, margin sampling doesn't necessarily gives it because it discards other classes' informations. So as a more general strategy, uncertainty query should calculate the *entropy* of the posterior probabilities which can be shown as:

$$B^* = \underset{B_i \in B^{\{U\}}, i=1 \dots N^{\{U\}}}{\operatorname{argmax}} -\sum_{j=1}^{C} P_\theta(y_j|B_i) \log P_\theta(y_j|B_i) \tag{4.3}$$

where C is the number of class labels in the dataset.

Note that for binary classification problems this entropy-based query strategy will yield to the same result with margin and uncertainty strategies. But the entropy based approach generalizes easily to probabilistic multi-label classifiers [31].

4.2 Multiple Instance Active Learning

In a single instance learning problem, active learning strategies are developed over the years but for MIL framework, it is not straightforward to apply these strategies. Because, the datasets are constructed as bag of instances so querying becomes a

confusing task. To address this problem, Settles et al [33] proposed the first Multiple Instance Active Learning strategy where they query the some of the instances in a selected bag. This would allow us to label just one instance instead of a whole bag of instances.

Their approach is useful for some applications where the instance labeling can be done. But for some applications, labeling an instance is not an easy task for human experts. To solve this problem, Fu et al. [34] adopted Fischer Information Matrix based query strategy for bag-level active learning. In their work, Fisher Information Matrix is constructed for the bags and the query is done by selecting the smallest fisher score.

Alternatively, to address this problem Liu et al. [35] proposed a kernel-based MI active learning. In their work, they used set kernels to calculate the distance between two bags. After the set kernel values are calculated, they trained an SVM classifier. As for the active learning, they proposed an *MI Informativeness* measure. In the informativeness measure, they combined three different metrics which utilize the kernel values of the bags. Those metrics are *uncertainty*, *novelty* and *diversity* measures. Uncertainty is the same measure that has given in equation 4.1 but in Liu et al. work, they used SVM classifier so instead of posterior probability, they used the objective function of SVM. So the uncertainty measure of a given data sample would be as the following:

$$u\left(B_j^{\{U\}}\right) = 1 - \left|\sum_i^{N^{\{T\}}} \eta_i K_{nset}\left(B_j^{\{U\}}, B_i^{\{T\}}\right) + b\right| \tag{4.4}$$

where η_i is the Lagrangian multiplier of the support vectors.

The second metric for informativeness measure is the novelty measure. Intuition of the novelty measure is that unlabeled data sample which is close to the training data will be less likely to be queried. In this case, unlabeled data sample which is far away from the training data will have a higher chance to be selected. This approach allows us to improve our classifier model by selecting data sample with minimum overlapping with the training data. Novelty measure of an unlabeled data $B_j^{\{U\}}$ can be given as,

$$d\left(B_j^{\{U\}}\right) = 1 - \max_{1<i<N^{\{T\}}} K_{nset}\left(B_j^{\{U\}}, B_i^{\{T\}}\right) \tag{4.5}$$

Lastly, the third metric is the *diversity* measure. This measure calculates the diversity of a given unlabeled data sample between the whole unlabeled dataset. This metric prevents the query possibility of two almost the same unlabeled data samples. Diversity measure can be given as:

$$r\left(B_j^{\{U\}}\right) = 1 - \sum_{i=1, i\neq j}^{N^{\{U\}}} K_{nset}\left(B_i^{\{U\}}, B_j^{\{U\}}\right) / \left(N^{\{U\}} - 1\right) \tag{4.6}$$

So far, we measured the proximity with the decision boundary with uncertainty measure, the diversity between training and unlabeled data sample with novelty measure and diversity among the unlabeled data with diversity measure. The combination of these three measurements would yield the MI Informativeness measurement which is given as follows:

$$\begin{aligned} &MI\_Informativeness\left(B_j^{\{U\}}\right) \\ &\quad = \lambda \times u\left(B_j^{\{U\}}\right) + (1-\lambda) \times r\left(B_j^{\{U\}}\right) \times d\left(B_j^{\{U\}}\right) \end{aligned} \tag{4.7}$$

where λ is a trade-off parameter between metrics. After calculating the informativeness measure for every unlabeled data sample, unlabeled data sample which gives the maximum value is queried and added to the training data.

4.3 Dictionary Ensemble Based Multiple Instance Active Learning

As stated in the previous section, there are some studies on combining the active learning techniques and MIL framework but these algorithms don't benefit the advantages of sparse coding and classifier ensembles. So, we propose an MI active learning technique that uses sparse coding and classifier ensemble methods – as we call it DEMIAL. Overview of the proposed algorithm is shown in Figure 4.1. The DEMIAL algorithm combines SCCE-MIL and active learning and extends it by using different base classifiers and different active learning query strategies.

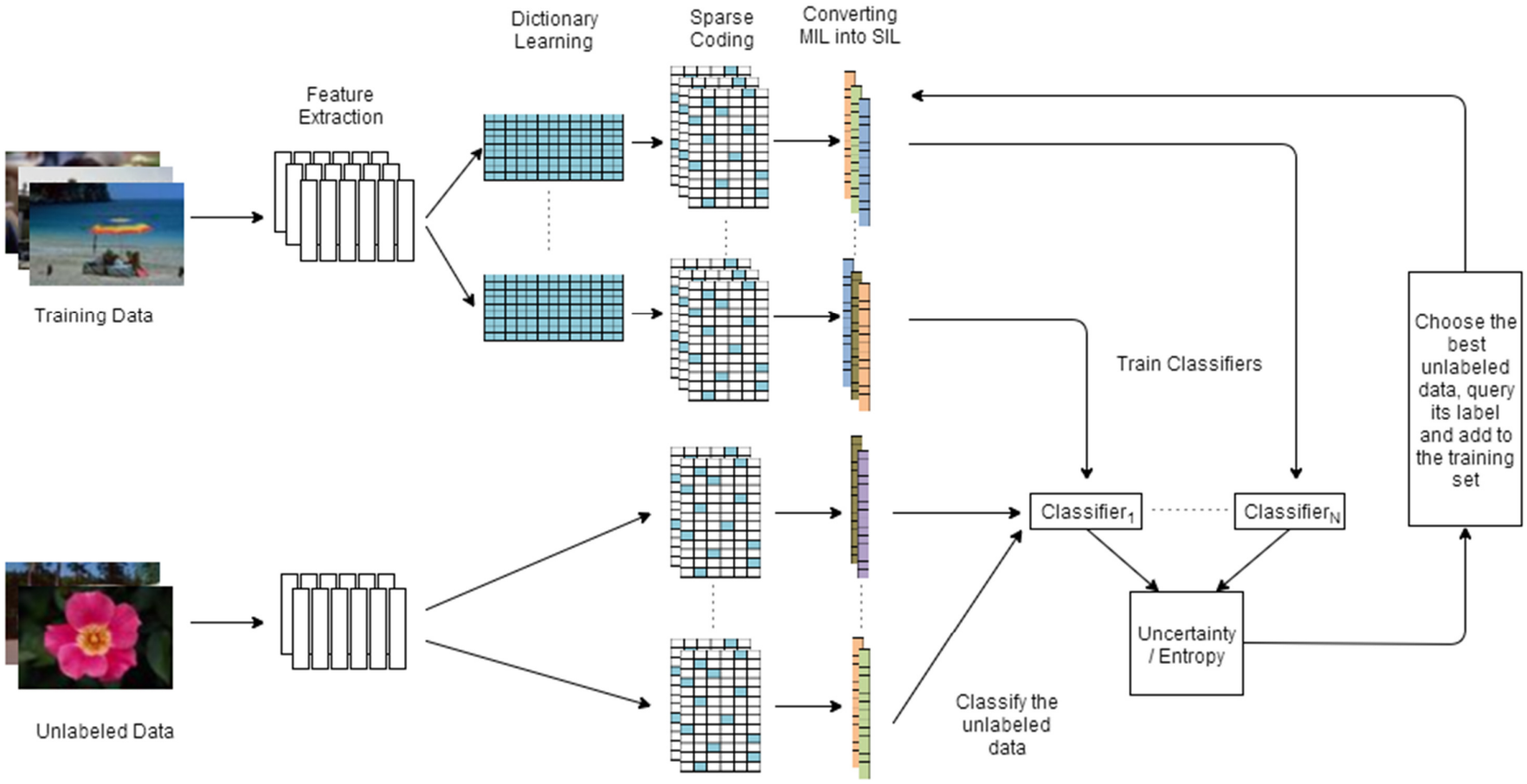

Figure 4.1 : Overview of the proposed DEMIAL algorithm.

Fundamentally, our approach consists of a training stage and active learning stage. In the training stage, there are four consecutive steps to model the data. Firstly, we learn different sized dictionaries. Multiple dictionaries allow us to utilize the ensemble strategy and model every attributes of the given signal. With this approach, we are building an ensemble system with different sized dictionaries.

In the next step, we obtain the sparse representation of the training data by using the learned dictionaries. In each dictionary, a training signal has a different representation, meaning that different atoms in the dictionary are used for the representation. While varied sized dictionaries capture the different patterns in the training signals, by calculating the sparse representations noise in the data is eliminated and signals are well represented.

When the sparse representations are calculated for each training data, we can construct the sparse bag of instances. In MIL, the dictionaries are trained in instance space and each instance is transformed into sparse feature vector using these dictionaries. Once the sparse instance features are calculated, we can unite all the sparse features and obtain the sparse bag features of the training data.

As we mentioned in Section 2.1, there are several approaches for solving a MIL problem. One of which is the embedded-space algorithms particularly the pooling functions. Since pooling functions give us the statistical summary of a given bag and the sparse features give the patterns of each instance in the bag, this function becomes suitable for our algorithm. Among the pooling functions, the max pooling function gives better results when used with sparse coding [29]. Each sparse bag is converted into a single bag feature by using the pooling function. Then we use a base classifier e.g. SVM, MLP and DT to model the bag features.

Once the training stage is done, we can utilize the unlabeled data with active learning stage. In active learning stage, each data sample in the unlabeled dataset is represented with sparse features and sparse instance features are converted into a single bag feature. In the sparse coding stage, the trained dictionaries which are constructed using training dataset are used. Next the bag features of the unlabeled data are classified using base classifiers. Then, we combine the class results of the ensemble system by using uncertainty/entropy measures. Once the measurement values are computed, the best informative data sample is selected for the query

process. After that, selected unlabeled data sample and its given label by the oracle is added to the training set. This process is continued until a pre-determined number of iterations. When the active learning is done, the trained model is used for testing. In testing, the decision of the ensemble is calculated by using the majority voting. Pseudo-code of training and active learning stage of the proposed DEMIAL algorithm is shown in Table 4.1.

Table 4.1 : DEMIAL algorithm.

Input: Training set $B^{\{T\}}$, Unlabeled set $B^{\{U\}}$, dictionary sizes $S = \{s_1, \dots, s_L\}$, ensemble size L, Number of active learning rounds m
Output: Trained set of classifiers
1. Concatenate all the instances in the bags and construct the instance matrix $\tilde{b} = \{b_1, \dots, b_M\}$ where $M = \sum_i^{N^{\{T\}}} K_i$.
2. for i=1: L
3. Train an *S(i)* sized dictionary.
4. for j=1:size($B^{\{T\}}$)
5. Calculate the sparse representations of every instance in the *jth* bag
6. Calculate the bag feature $\bar{B}_{ij}^{\{T\}}$ using the *max pooling function*
7. end
4. for j=1:size($B^{\{U\}}$)
5. Calculate the sparse representations of every instance in the *jth* bag
6. Calculate the bag feature $\bar{B}_{ij}^{\{U\}}$ using the *max pooling function*
7. end
8. end
9. for k=1:m
10. for i=1: L
11. Train classifier$_i$ using bag features $\bar{B}_i^{\{T\}} = \left\{\bar{B}_{i1}^{\{T\}}, \dots, \bar{B}_{ij}^{\{T\}}, \dots\right\}$
12. Predict label of the $\bar{B}_i^{\{U\}}$ using classifier$_i$
13. end
14. Calculate the entropy/uncertainty measure and select the instance $\bar{B}^*$ which maximize the entropy/uncertainty.
15. Query B^* its label to the oracle and add $\langle B^*, y^* \rangle$ tuple to training set.
16. end

4.4 Experimental Results

Experiments for the DEMIAL algorithm are conducted on COREL 1000, 2000 [11], Elephant, Fox and Tiger [6] datasets. Details of these datasets are explained in Section 3.2. We tested our algorithm in two experiment. In each experiment, we divided into three different subsets: training, unlabeled and test. Each dataset is divided into 5 folds and each subset is constructed using these folds with the ratio of

20%, 60% and 20% of training, unlabeled and test respectively. In both of the experiments, initial training is made with the training subset. Later, the active learning is conducted in the unlabeled subset. The final classifier is tested with the test subset.

In the first experiment we compare the kernel based MI active learning method with MI informativeness active learning query technique proposed by Liu et al. [35] and the proposed DEMIAL approach with uncertainty (U) and entropy (E) measures. In the second experiment, we compare the effect of the base classifiers with different query strategies in DEMIAL algorithm. These experiments are detailed in the next subsections.

4.4.1 Comparing with the kernel-based MI active learning method

Kernel-based MI active learning method is proposed by Liu et al. [35] in which the distances of the bags are calculated using set kernel function and SVM is trained with these distances. In the active learning phase, they proposed an MI informativeness measure which utilizes three measures and combines them – this method is explained previously in Section 4.2. As the kernel-based MI active learning algorithm uses the SVM classifier, in our first experiment we compared it with the proposed DEMIAL algorithm using SVM as a base classifier. Besides we also compare the DEMIAL active learning query strategies using uncertainty and entropy measures. The results of the experiments are shown in Figure 4.2.

As it shown in the figures, increasing the training data size with active learning increases the classification accuracies for most of the datasets. In Fox and Tiger, the improvement varies compared to the other datasets. If we compare the algorithms, we can see that the proposed DEMIAL algorithm performs better than the kernel-based MI active learning method. We denote that the initial training size is the 20% of the whole dataset and the active learning is continued until 50% of the data is in the training set. The final results of these algorithms are similar to the results obtained without using active learning experiments for some datasets. On top of that, we achieve these results with less training data. Note that none of the algorithms control the labels of the queried data samples. If all of the queried data samples are labeled with the same class label, this may affect the balance of the training dataset and may radically change the decision boundary of the classifier at the next iteration.

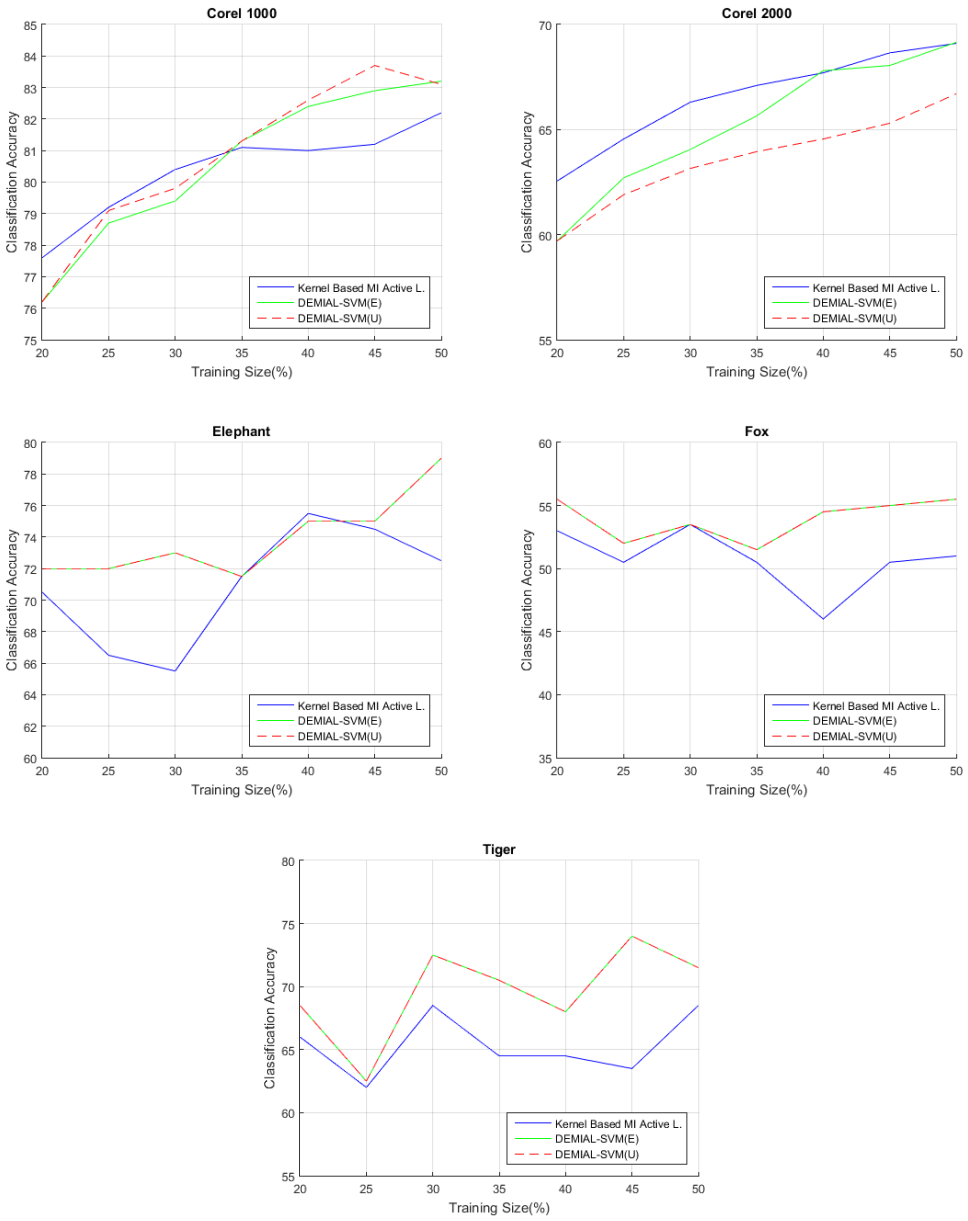

Figure 4.2 : Comparison between DEMIAL and kernel based MI active learning algorithm.

In addition, if you compare the query strategies in the DEMIAL method, most of the accuracy results are similar. In fact, the results of the Elephant, Fox and Tiger datasets are exactly the same. This is because these three datasets are binary classification problems and both query strategies choose the same data sample in the query stage. But in Corel 1000 and 2000, the results are close with each other. In order to find the best combination, we will investigate the effect of the query strategy with different base classifiers in the next subsection.

4.4.2 Comparing the base classifiers with different query strategies

One of the superiority of the proposed DEMIAL algorithm to kernel based MI is the ability to use different base classifiers. Selecting the most informative data sample for the base classifier is also another important issue for the DEMIAL algorithm. So, the following experiments are carried out to investigate the effect of the query strategy with different base classifiers in the proposed algorithm. Since the MI informativeness measure requires kernel information between two bags, this measure is only suitable for SVM classifier and we cannot adopt it to the DEMIAL algorithm. The classification accuracies for the DEMIAL algorithm with different base classifiers for entropy and uncertainty query strategies are given in Figure 4.3.

In terms of the base classifiers, we can see different performances for each dataset. For example, in Corel 1000 and 2000 datasets, SVM is significantly better than DT and MLP. However, in Elephant and Tiger, it performs poorly. For the Fox dataset, neither different base classifiers nor different query strategies produce significant results. This is partly because of the dataset itself. In some previous works, Fox

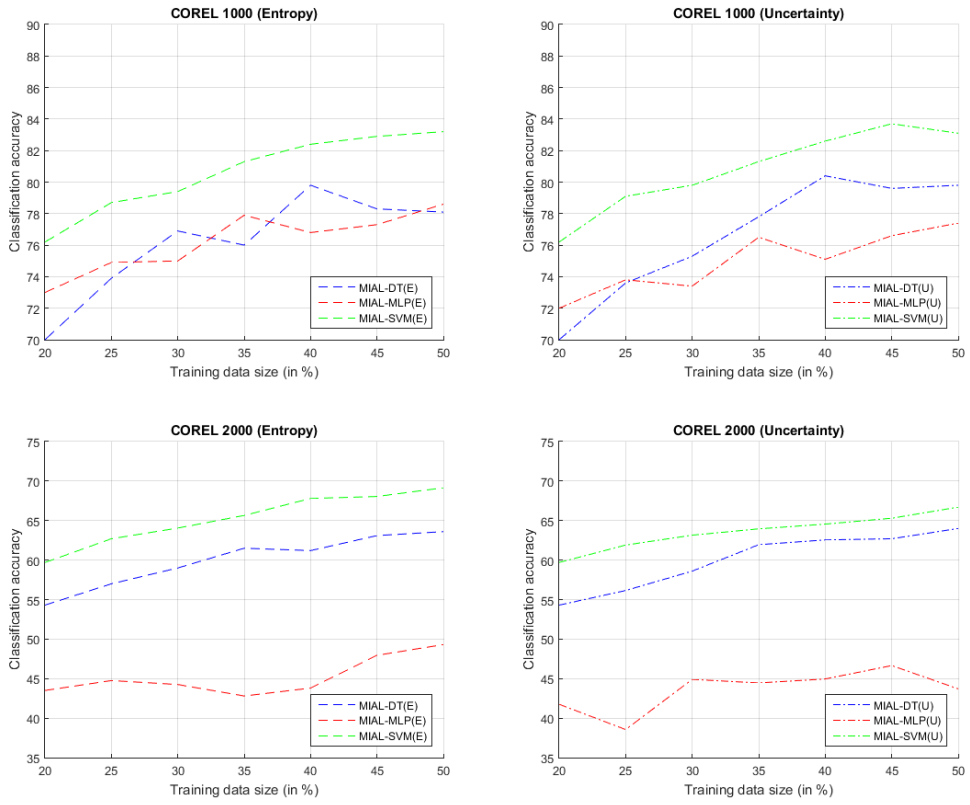

Figure 4.3 : Comparison of the base classifiers in DEMIAL algorithm with *entropy* and *uncertainty* query strategy.

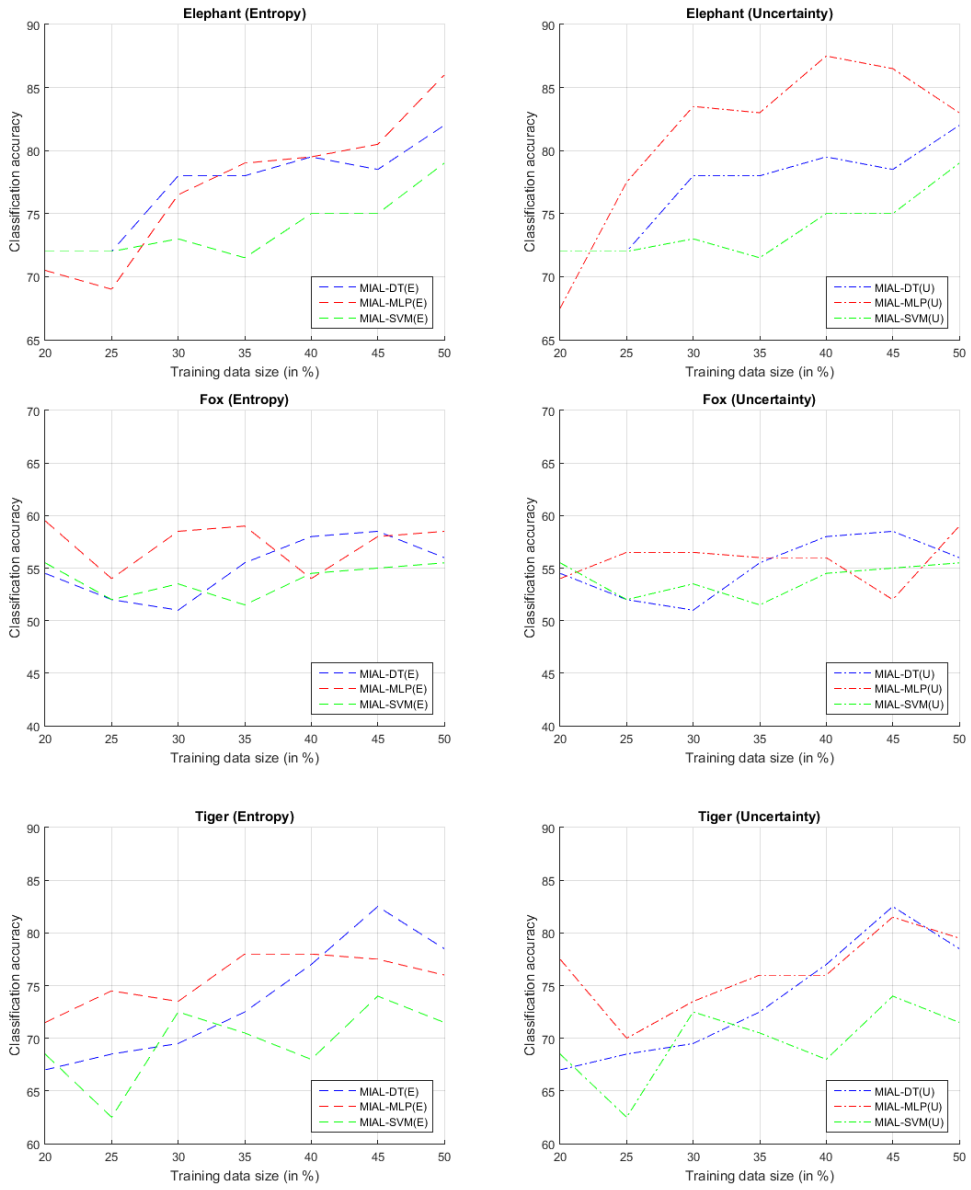

Figure 4.3 (continued): Comparison of the base classifiers in DEMIAL algorithm with *entropy* and *uncertainty* query strategy.

dataset doesn't produce good results in terms of active learning [36] and classifier ensemble strategy [37].

In addition, if we compare the active learning query strategies, for Elephant, Fox and Tiger datasets uncertainty and entropy measures give the same results using DT and SVM because they query the same data samples. As MLP starts with random initial weights it gives slightly different results when we use different query strategies.

We also conducted pairwise t-tests on the experimental results and the p-values with respect to base classifiers and query strategies are given in Table 4.2. The significant

difference between paired classifiers ($p < 0,05$) are shown in bold. From these results we can state that the significant accuracy differences between classifiers can be seen for Corel 1000 and Corel 2000 datasets. If we consider the superiority of the selection strategies for each base classifier the significant difference between entropy and uncertainty can be seen for Corel 2000 dataset. In addition, the only significant difference between algorithms for Tiger can be seen between MLP-U and MLP-E.

Table 4.2 : p-values for DEMIAL algorithm with respect to base classifier and query strategies using paired t-test.

	Corel 1000	Corel 2000	Elephant	Fox	Tiger
DT-E / DT-U	0,087	0,686	1	1	1
DT-E / MLP-E	0,669	**0,003**	0,178	0,413	0,142
DT-E / MLP-U	0,525	**<0,001**	0,749	0,208	0,688
DT-E / SVM-E	**0,004**	**0,008**	0,709	0,871	0,221
DT-E / SVM-U	**0,013**	**0,007**	0,709	0,871	0,221
DT-U / MLP-E	0,329	**0,004**	0,178	0,413	0,142
DT-U / MLP-U	0,176	**<0,001**	0,749	0,208	0,688
DT-U / SVM-E	**0,003**	**0,003**	0,709	0,871	0,221
DT-U / SVM-U	**0,008**	**0,043**	0,709	0,871	0,221
MLP-E / MLP-U	0,438	0,150	0,324	0,749	**0,025**
MLP-E / SVM-E	**0,016**	**0,002**	0,351	0,448	0,346
MLP-E / SVM-U	**0,005**	**0,003**	0,351	0,448	0,346
MLP-U / SVM-E	**0,022**	**<0,001**	0,630	0,311	0,099
MLP-U / SVM-U	**0,021**	**<0,001**	0,630	0,311	0,099
SVM-E / SVM-U	0,900	**0,035**	1	1	1

5. CONCLUSION AND FUTURE WORK

In this thesis, we cover some of the prominent algorithms that are used to solve MIL problem and propose an MI active learning algorithm that uses sparse coding and classifier ensemble techniques. The proposed DEMIAL algorithm is compared with the kernel-based MI active learning algorithm. Besides query strategy which is one of the main factors in the success of active learning framework is also analyzed in the scope of this thesis. In the experimental results we implemented two different query strategies with different base classifiers and compared them on various MIL datasets.

It turned out that the proposed algorithm has better classification accuracy over the kernel-based MI active learning strategy. The reason is that the proposed algorithm utilizes the representative ability of the sparse coding and the generalization effect of the ensemble strategy. These methods elaborate patterns of the signal which allows training better classifiers.

Except for few cases classifiers gained from unlabeled examples with active learning. From the experimental results it turns out that the best base classifier and query strategy is dependent to the problem space that we are dealing with.

On the other hand, selection of base classifier in our proposed algorithm has variable results for different datasets. This variation is not seen in the multiclass Corel 1000 and 2000 datasets. However, it is an issue for Elephant, Fox and Tiger binary class datasets. This can be caused from the class inequality after the active learning iteration. As a future work, we will investigate the issues behind the variance of the results by comparing with more datasets and more query strategies. Besides we will integrate diversity, uncertainty and novelty based sampling strategies to DEMIAL algorithm.

REFERENCES

[1] **Dietterich, T. G., Lathrop, R. H., and Lozano-Pérez, T.** (1997). Solving the multiple instance problem with axis-parallel rectangles. *Artificial intelligence 89.1*, 31-71.

[2] **Amores, J.** (2013). Multiple instance classification: Review, taxonomy and comperative study. *Artificial Intelligence 201*, 81-105.

[3] **Maron, O., Lozano-Pérez, T.** (1998). A framework for Multiple-Instance Learning. In *Advances in Neural Information Processing Systems 10*, 570-576.

[4] **Babenko, B.** (2008). Multiple instance learning: algorithms and applications. *View Artic. PubMed/NCBI Google Sch.*, 1-19.

[5] **Zhang, Q., and Goldman, S.A.** (2002), EM-DD: An improved multiple-instance learning technique. In *Advances in Neural Information Processing Systems 14*, 1073-1080.

[6] **Andrews, S., Tsochantaridis, I., and Hofman, T.** (2003), Support vector machines for multiple-instance learning. In *Advances in Neural Information Processing Systems 15*, 561-568.

[7] **Wang, J., and Zucker, J.-D.** (2000), Solving multiple-instance problem: A lazy learning approach. In *Proc. 17th International Conf. on Machine Learning*, 1119-1125.

[8] **Zhang, J., Marszalek, M., Lazebnik, S., and Schmid, C.** (2007), Local features and kernels for classification of texture and object categories: A comprehensive study, *International Journal of Computer Vision 73.2*, 213-238.

[9] **Gärtner, T., Flach, P.A., Kowalczyk, A., and Smola, A.J.** (2002), Multi-instance kernels. In *Proc. 19th International Conf. on Machine Learning*, 179-186.

[10] **Chen, Y., and Wang, J.Z.** (2004), Image categorization by learning and reasoning with regions. *The Journal of Machine Learning Research 5*, 913-939.

[11] **Chen, Y., Bi, J., and Wang, J.Z.** (2006), MILES: Multiple-instance learning via embedded instance selection. *Pattern Analysis and Machine Intelligence, IEEE Transactions 28.12*, 1931-1947

[12] **Polikar, R.** (2006), Ensemble based systems in decision making. *Circuits and systems magazine, IEEE 6.3*, 21-45.

[13] **Kuncheva, L.I.** (2004), *Combining Pattern Classifiers: Methods and Algorithms.* John Wiley & Sons.

[14] **Schapire, R.E.** (1989), The strenght of weak learnability. *Machine Learning 5.2*, 28-33.

[15] **Freund, Y., and Schapire, R.E.** (1997), A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences 55.1*, 119-139.

[16] **Ho, T.K.** (1998), The random subspace method for consturcting decision forests. *Pattern Analysis and Machine Intelligence, IEEE Transactions 20.8*, 832-844.

[17] **Lee, H., Battle, A., Raina, R., and Ng, A. Y.** (2006), Efficient sparse coding algorithms. In *Advances in Neural Information Processing Systems*, 801-808.

[18] **Elad, M.** (2010), *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*. Springer US.

[19] **Mallat, S.G., and Zhang, Z.** (1993), Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing 41.12*, 3397-3415.

[20] **Pati, Y.C.C., Rezaiifar, R., and Krishnaprasad, P.S.S.** (1993), Orthogonal matching pursuir: recursive function approximation with applications to wavelet decomposition. *Proceedings of the 27 th Annual Asilomar Conference on Signals, Systems, and Computers*, 1-5.

[21] **Donoho, D.L.** (2006), For most large underdetermined systems of linear equations the minimal l1-norm solution is also the spartest solution. *Communications on pure and applied mathematics 59.6*, 797-829.

[22] **Chen, S.S., Donoho, D.L., and Sounders, M.A.** (1998), Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing 20.1*, 33-61.

[23] **Efron, B., Hastie, T., Johnstone, I.M., and Tibshirani, R.** (2004), Least angle regression. *The Annals of statistics 32.2*. 407-499.

[24] **Engan, K., Aase, S.O., and Husoy, J.H.** (1999), Method of optimal directions for frame design. *Acoustics, Speech, and Signal Processing, 1999. Proceedings 5*, 2443-2446.

[25] **Aharon, M., Elad, M., and Bruckstein, A.** (2006), K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *Signal Processing, IEEE Transactions on 54.11*, 4311-4322.

[26] **Marial, J., Bach, F., Ponce, J., and Sapiro, G.** (2009), Online dictionary learning for sparse coding. In *Proceedings of the 26th annual international conference on machine learning*, 689-696.

[27] **Song, X., Jiao, L.C., Yang, S., Zhang, X., and Shang, F.** (2013), Sparse coding and classifier ensemble based multi-instance learning for image categorization. *Signal Processing 93.1*, 1-11.

[28] **Marial, J., Bach, F., Ponce, J., Sapiro, G., and Zisserman, A.** (2008), Supervised dictionary learning. *Advances in Neural Information Processing Systems*¸1033-1040.

[29] **Yang, J., Yu, K., Gong, Y., and Huang, T.** (2009), Linear spatial pyramid matchin using sparse coding for image classification. In *Computer*

Vision and Pattern Recognition, 1794- 1801.

[30] **Lewis, D.D., and Gale, W.A.** (1994), A sequential algorithm for training text classifiers. *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 3-12.

[31] **Settles, B.** (2010), Active learning literature survey. *University of Wisconsin, Madison 52*.

[32] **Scheffer, T., Decomain, C., and Wrobel, S.** (2001), Active hidden markov models for information extraction. *Advances in Intelligent Data Analysis*, 309-318.

[33] **Settles, B., Craven, M., and Ray, S.** (2008), Multiple-instance active learning. *Advances in Neural Information Processing Systems 20.1*, 1289-1296.

[34] **Fu, J., and Yin, J.** (2011), Bag-level active multi-instance learning. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2011 Eighth International Conference 2*, 1307-1311.

[35] **Liu, D., Hua, X., Yang, L., and Zhang, H.** (2009), Multiple-instance active learning for image categorization. In *Advances in Multimedia Modeling*, 239-249.

[36] **Zhang, D., Wang, F., Shi, Z., and Zhang, C.** (2010), Interactive localized content based image retrieval with multiple-instance active learning. *Pattern Recognition vol. 43-2,* 478-484.

[37] **Cheplygina, V., Tax, D.M.J., and Loog, M.** (2015), Dissimilarity-based ensembles for multiple instance learning. *IEEE Transactions on Neural Networks and Learning Systems vol. 27-6*, 1379-1391

[38] **Cortes, C., and Vapnik, V.** (1995), Support-vector networks, *Machine Learning 20.3*, 273-297.

[39] **Alpaydın, E., and Bishop, C.M.** (2014), *Introduction to Machine Learning.* MIT Press.

[40] **Kuhn, H.W., and Tucker, A.** (1951), Nonlinear Programming, In *Proceedings of the Second Symposium on Mathematical Statistics and Probability*, 481-492.

APPENDICES

APPENDIX A: Base Classifiers

APPENDIX A

APPENDIX A.1 SUPPORT VECTOR MACHINES

Support Vector Machines (SVM) is a supervised learning algorithm proposed by Cortes and Vapnik [38]. In SVM, each data sample is mapped into a high-dimensional feature space so that a non-linearly separable dataset would turn into a linearly seperable data. After the mapping, a high-dimensional decision boundary is constructed with a margin. We want the margin as largest as possible while selecting the minimum margin between decision boundary and training examples. The nearest training examples are called support vectors and the decision boundary is a linear combination of the support vectors. Figure A.1 shows the support vectors, margin and decision boundary [39].

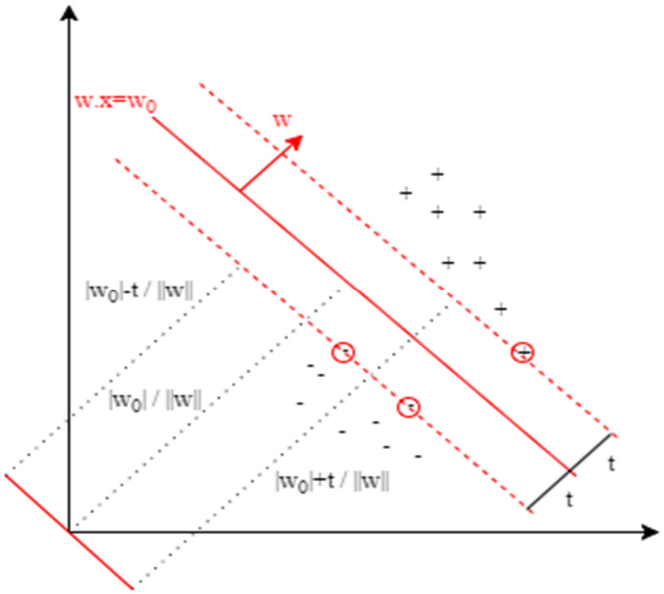

Figure A.1 : Example of support vectors, decision boundary and margins [39].

Mathematically, suppose that all the data satisfy the constraint

$$y_i(w \cdot x_i + w_0) \geq 1 \quad \forall i \tag{A.1}$$

where w is the normal to the decision boundary and $|w_0|/\|w\|$ is the distance between decision boundary and origin. In order to minimize the distance between data sample and decision points, we need to minimize the margin by solving the

following optimization problem:

$$\min \frac{1}{2}\|w\|^2 \quad s.t. \quad y_i(w \cdot x_i + w_0) \geq 1 \quad \forall i \tag{A.2}$$

If we apply the Lagrangian formulation, equation A.2 would turn into

$$\min \frac{1}{2}\|w\|^2 - \sum_{i=1}^{N} \eta_i y_i (w \cdot x_i + w_0) + \sum_{i=1}^{N} \eta_i \tag{A.3}$$

where $\eta_i \geq 0$ is a Lagrange multiplier. Applying the Karush-Kuhn-Tucker (KKT) conditions [40] and substitute to equation A.3 , we will have the following dual optimization:

$$\max_{\eta} \sum_i \eta_i - \frac{1}{2} \sum_{i,j} \eta_i \eta_j y_i y_j x_i \cdot x_j \quad s.t. \quad \forall i \begin{cases} \sum_i \eta_i y_i = 0 \\ \eta_i > 0 \end{cases} \tag{A.4}$$

In this dual optimization problem, most of the η_i are zero. The non-zero ones are called support vectors.

The above formulations are viable for the linearly separable dataset but what about a non-separable one? Since we cannot learn a non-linear SVM classifier, we can map the dataset into a new space using a non-linear transformation. This can be done with a kernel function which can be shown as

$$K(x_i, x_j) = \phi(x_i)\phi(x_j) \tag{A.5}$$

where $\phi(x)$ is the mapping function. We can use any kernel function for SVM. Most popular kernel functions are polynomial, radial-basis function, etc. If we incorporate the kernel function into our dual optimization function, we will have

$$\max_{\eta} \sum_i \eta_i - \frac{1}{2} \sum_{i,j} \eta_i \eta_j y_i y_j \phi(x_i)\phi(x_j) \quad s.t. \quad \forall i \begin{cases} \sum_i \eta_i y_i = 0 \\ \eta_i > 0 \end{cases} \tag{A.6}$$

APPENDIX A.2 MULTI-LAYER PERCEPTRON

Multi-layer Perceptron (MLP) is an Artificial Neural Network (ANN) model that can be used for regression and classification. Perceptrons are used in MLP as a model of human brain neurons. Perceptron is the basic processing unit which has number of inputs and one output. It calculates the weighted sum of the inputs. If the sum exceeds the threshold, it produces the output true. If it doesn't exceed, it produces false. If we think this as a linear discriminant function, a single perceptron can classify binary classification problem. For C-class problem, we need C parallel perceptrons.

Although perceptron can solve a linear classification problem, it needs different approach for solving a non-linear problem. It can be achieved by adding an intermediate level of perceptrons (or called hidden layer) so that second layers of perceptrons input comes from the outputs of first layers of perceptron, thus MLP is constructed and this can be seen in Figure A.2 [39].

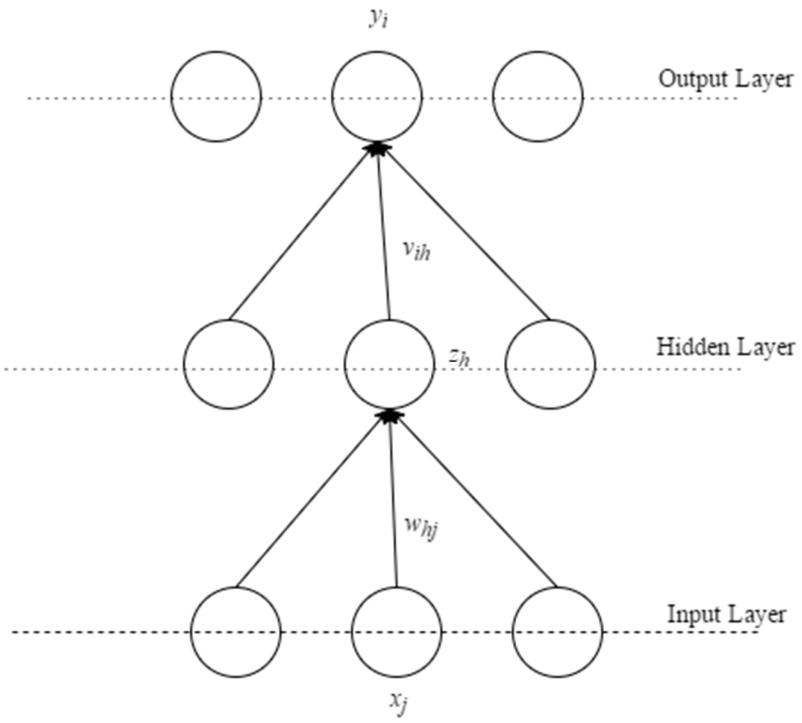

Figure A.2 : Example of a multi-layer perceptron with one hidden layer [39].

Output of the MLP is the same as the perceptrons. If we apply to a binary classification, we need one output node. If we apply to a C-class problem, we need C

output node. Having more than one layer of perceptrons can be applied and this would let us learn more complex structures in the dataset which is called deep learning but this is beyond the scope of this brief explanation.

APPENDIX A.3 DECISION TREES

Decision Tree (DT) is a non-parametric estimation technique which recursively splits the input space into smaller local regions and combines them in a hierarchical model. DT is made of internal decision nodes, which provides a decision function that results a discrete output, and leaf nodes which is the result of the class value or numerical value. The illustrative example is given in Figure A.3 [39]. This approach allows us to break down a complex problem into a set of simpler decisions. Another advantage of the DT is to calculate faster decisions because of its hierarchical structure of the decision functions where at each step of decision, the classifier reduces the half of the results.

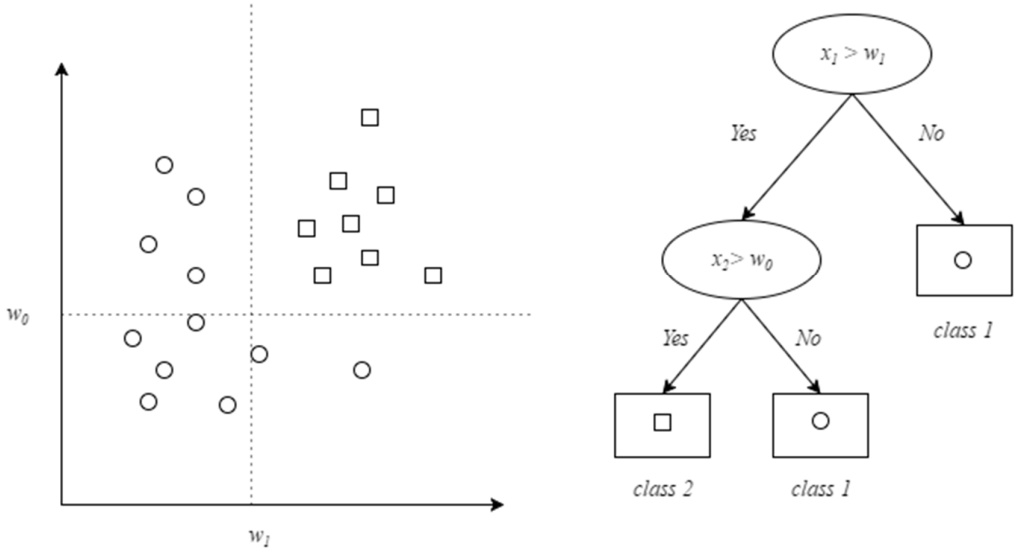

Figure A.3 : Example of a decision tree with the dataset [39].

CURRICULUM VITAE

Name Surname **: Gökhan KOÇYİĞİT**

Place and Date of Birth **: İSTANBUL, 18.11.1987**

E-Mail **: kocyigitg@itu.edu.tr**

EDUCATION **:**

- **B.Sc.** **:** 2009, Turkish Naval Academy, Comptuer Engineering

PUBLICATIONS, PRESENTATIONS AND PATENTS ON THE THESIS:

- Kocyigit, G. and Yaslan, Y., 2016, Dictionary Ensemble based Multi Instance Active Learning Method for Image Categorization, *Signal Processing and Communications Applications Conference (SIU), 2016 24th*. IEEE.